Austin Cody 1/31/2024

NBA Salary Predictor and Team Optimizer

The Backstory

The National Basketball Association had 467 active players in the 2022-2023 season, each varying immensely in skill, experience, and salary. Given a team's budget, what is the best possible combination of 15 players to make up a team while keeping a balance of player positions? The most succinct measure of player value is their current salary, but players inevitably over/underperform after their salary is designated. We can quantify this over/underperformance by using machine learning techniques to predict a player's value based on a multitude of performance statistics in a season, and then use linear programming to find an optimal team combination.

My Bayesian optimized random forest regressor model proved to be the strongest of the tested models with a cross-validated r^2 of 0.72. This model can be used to identify an optimal team combination for any team or to simply identify over/undervalued players as targets for a trade.

Data Wrangling

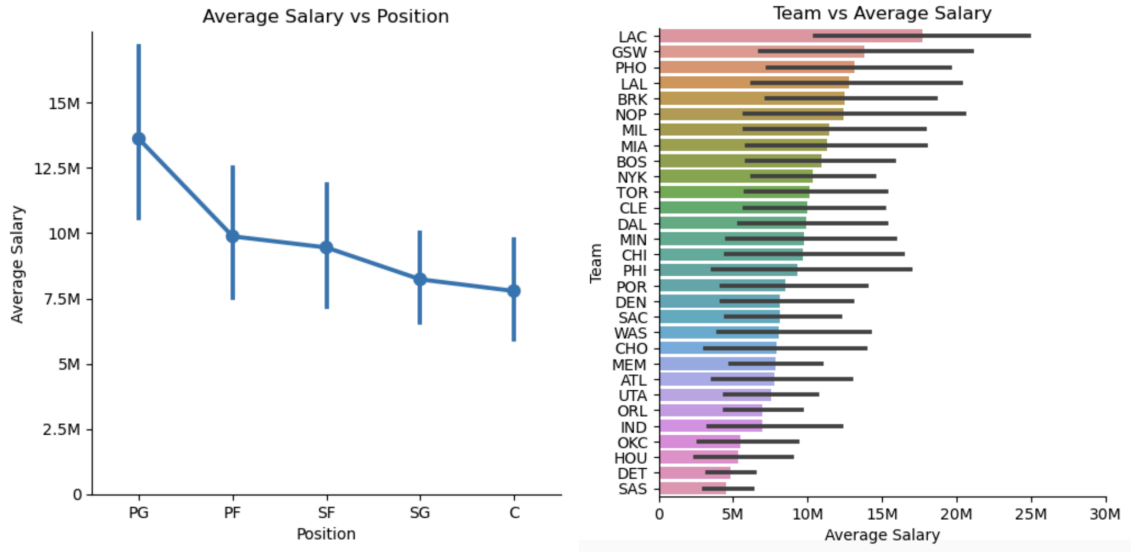
The csv I pulled from Kaggle was scraped from Basketball Reference. After converting to a Pandas DataFrame, it contained 467 rows with 31 columns.

- Problem 1 : Two columns contained irrelevant information.
    - Solution: Two columns were dropped. The remaining columns were name, salary, position, team, age, followed by 24 player statistics columns.
- Problem 2: Each statistic column was named as an abbreviation of its statistic. For example, total rebounds per game was represented by "TRB" and effective field goal percentage was represented by "eFG%". At the time I did not possess the domain knowledge to identify each column.
    - Solution: I renamed all such columns to be more readable.
- Problem 3: The two categorical columns are position and team, however some players played for multiple positions and multiple teams. These were stored as strings with the team abbreviations joined by a "/" and the position abbreviations joined by a "-". This could cause problems later on in the creation of dummy variables.
    - Solution: I converted each value in the columns to list objects containing 1 or more team/position. This did in fact save time in the preprocessing step.
- Problem 4: The only columns that had missing values were percentage columns. Examining these columns showed that all were the result of 0 division, from players who did not attempt a particular type of shot. In most cases this was because a player had a small number of minutes per game or games played.

- Solution: Convert null values to 0 and drop rows with fewer than 10 games played.
- Note: There were 7 players who had a high number of games played and had 0 3-point attempts resulting in 0 division. I thought this was an error at first, but I noticed they all played the position Center. Centers typically hover close to the basket and infrequently shoot outside the key. It is reasonable that a successful Center would not have shot a 3-pointer so I left them in.

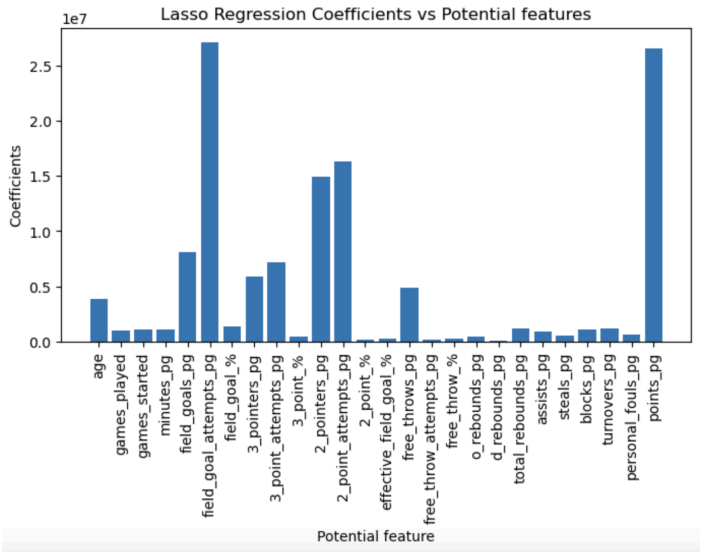
Exploratory Data Analysis

Examining the categorical variables position and team showed that some positions are valued more highly than others, and that some teams have significantly higher budgets.



For each numerical column I created a histogram to view the distribution of the data. This revealed that nearly every feature was skewed. I further examined the relationship between each numerical variable through a heatmap and a pair plot. The features that had a correlation coefficient with salary of 0.60 or higher were minutes_pg, field_goals_pg, 2_pointers_pg, free_throws_pg, assists_pg, and turnovers_pg.

To corroborate that these would be useful for machine learning, I used Lasso Regression (L1 Regularization) for feature selection. L1 Regularization adds an additional term to a linear model for each feature to reduce overfitting. In Lasso Regression, the coefficients of less predictive terms will converge to 0. Plotting these coefficients lets us see the most useful features:



This highlighted all shooting categories and age to be most useful as features for our machine learning models. Combining the features from examining the correlation coefficients of each variable and the features from L1 regularization provided me with the following features: age, 2_pointers_pg, 3_pointers_pg, free_throws_pg, assists_pg, points_pg, and minutes_pg. Note that to avoid multicollinearity, I removed the turnover column and each attempt column.

Data Preprocessing

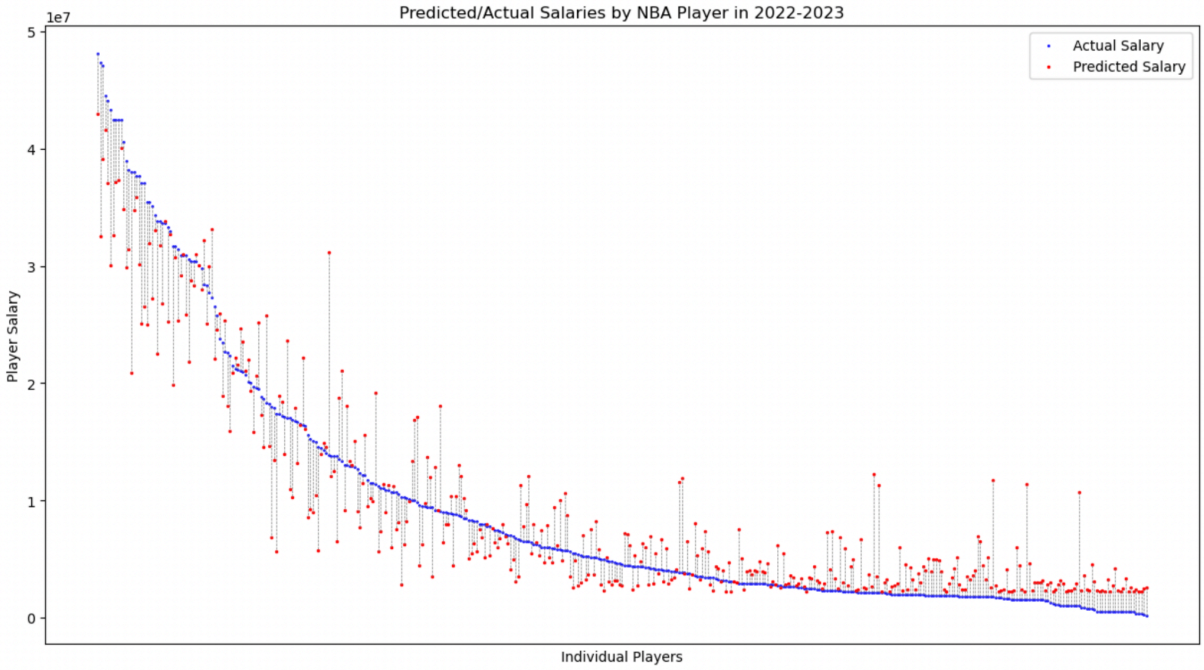
After creating dummy variables and splitting the data into a training set and a test set, I tested two different transformation techniques. I assumed that because so many of my selected features were right skewed that a PowerTransformer would handle the scaling better. I transformed the train and test data using both PowerTransformer and StandardScaler and used a linear regression model to evaluate their performance. The model using the power transformed data had an RMSE of $7,939,877 and an r^2 of 0.53. The model using the standard scaled data had an RMSE of $7,464,872 and an r^2 of 0.59.

I did not expect this result, but the model using the standard scaled data performed best, so I moved forward in this project using the StandardScaler.

Model Selection

I modeled using a Lasso Regression model, a Random Forest Regressor, and an XGBoost Regressor. For hyperparameter tuning I used GridSearchCV on the Lasso model, (because I was only tuning the alpha hyperparameter) and Bayesian Optimization for the Random Forest and XGBoost regressors. The Lasso model had an r^2 of 0.64. Both the Random Forest and XGBoost models had r^2 just under 0.72. The XGBoost model overfit to the training data to a greater extent than the Random Forest Regressor, so the final model selected was a Bayesian Optimized Random Forest Regressor.

Best Random Forest Regressor Hyperparameters:
max_depth: 7.046122532763487
max_features: 4.887569511250316
max_leaf_nodes: 29.644614290474742
min_samples_leaf: 1.6204560131132242
min_samples_split: 2.1333262991437074
n_estimators: 495.6940265294837



The above figure depicts the relationship between predicted salary and actual salary. My model slightly over-guesses on low-salaried players yet is underguessing on high-salaried players.

Team Optimizer

To find the optimal team I used PuLP for Linear Programming. The objective function to minimize was the salary prediction error, the difference between actual salary and predicted salary. The constraints were 15 players per team, a minimum of 2 players per position played for rotation and injury security, and a cumulative salary of the NBA salary cap for next season: $136,021,000.

The result was the following team:

name	salary	salary_predicted	salary_prediction_error
Harrison Barnes	18352273	25733256	-7380983
Lauri Markkanen	16475454	22170036	-5694582
Brook Lopez	13906976	31123767	-17216791
Jordan Clarkson	13340000	21091805	-7751805
Dillon Brooks	11400000	19185552	-7785552
Alec Burks	10012800	16860113	-6847313
De'Andre Hunter	9835881	17151845	-7315964
Mason Plumlee	9080417	18079033	-8998616
Jordan Poole	3901399	11555161	-7653762
Keldon Johnson	3873024	11888570	-8015546
Damion Lee	2133278	12221791	-10088513
Desmond Bane	2130240	11294308	-9164068
Tre Jones	1782621	11742386	-9959765
Austin Reaves	1563518	11392986	-9829468
Kris Dunn	1000001	10702841	-9702840

Total team cost: $118,787,882

Total salary prediction error: -$133,405,568

Our model estimates that this 15 person team is undervalued by over $130M while staying under budget and is balanced in terms of positions.

Future Research

This model could be generalized to players entering the NBA draft, or could be used to optimize a few open spots on a team while keeping others constant. To improve the accuracy of the model I could involve an injury likelihood predictor. Even forgoing the optimization section, this model can be used to simply identify how undervalued or overvalued a particular player is indicating that a player might be a good target for a trade.