Zachery Gebreab ZBG220000
CS 4375.003
Spring 2025

NBA Salary Prediction

1. Introduction

Although most casual NBA fans don't know about their favorite team's contract situations, the decisions made by front offices change teams forever. In a salary-capped league, players who don't provide enough value for what they are paid leave their team with a hole that cannot be filled because the team won't have much money left. A current example of this is with the Philadelphia 76ers and Paul George.

Right before the season started, the 76ers acquired George to a 4-year max contract deal, taking up a significant portion of their salary cap for close to the rest of the decade. Even though George has been considered a star for many years now, he is aging and has been suffering injuries, causing a drop in his performance far below . This season he is 35, played 41 games (only half of a full season), and averaged a modest 16.2 points, 5.3, rebounds, and 4.3 assists. His underperformance relative to how he was paid left his team with a 24-58 record, not even in playoff contention, and no money to help sign players. On top of that, other teams see this and will not be willing to trade for him, leaving the 76ers with no choice but to pay for his negligible impact for 3 more years.

This project dives deep into how player performance and other variables can be used to predict salaries for NBA players. By building a model for salary prediction, I hope to identify factors that influence contract value and help teams make data-driven decisions regarding allocation of their resources.

2. Dataset Overview and Preprocessing

2.1. Sources and Merging

Since data was collected for a variety of sources, proper preprocessing was a necessity to ensure consistency. For merging, player name and season were used together as a composite key to properly align everything like team record, salary, and stats. 5 main data frames were created, stats, salaries, awards, player info, and team record, and from there I could further merge everything. In order for the merging to take place, I had to standardize all of the datasets' column names (e.g changing all of the data sets' year-related columns to "season"") and normalize the data into a consistent format. After cleaning and joining, the biggest data frame, which included the stats, salary, awards, and team record data, had 9,052 rows and 76 columns.

2.2. Key Features

Our main data frames had many features to be explored. The stats data frame consisted of many features that you would expect like points, rebounds, assists, and steals. It also contained some less talked about stats like minutes, games started, and advanced stats like box plus-minus. The awards data frame had all the NBA accolades, and used binary indicators to track which player won the awards. The team record data frame had the teams' record along with how far they made it in the playoffs, and win percentage. The salary data frame contained our target variable, salary, along with the player name and year.

2.3. Cleaning and Handling Missing Data

To ensure a high quality EDA, several cleaning steps took place. One big thing I did was remove records where a player's salary was NA or 0, as this indicated the player was out of the league that season. Players who didn't earn any awards, which was the majority of players, were all given zeros for the various awards columns. I also had to convert a number of variables from strings into binary numbers so that they could be analyzed, like which all nba team a player was on or how far a team made it into the playoffs.

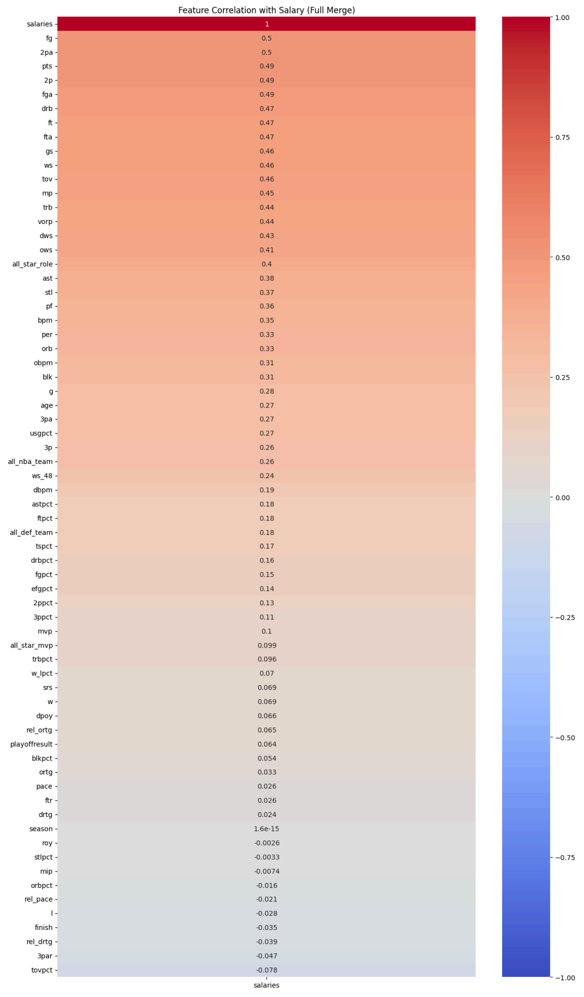
2.4. Feature Encoding and Transformation

Continuous features that changed over time, like salary, points, rebounds, and 3 pointers were normalized with z-score to account for the NBA handing out larger salaries and the game becoming more and more fast paced over the years. The transformation allows the model to better capture the proportional difference in salary instead of a flat dollar amount.

3. Exploratory Data Analysis (EDA)

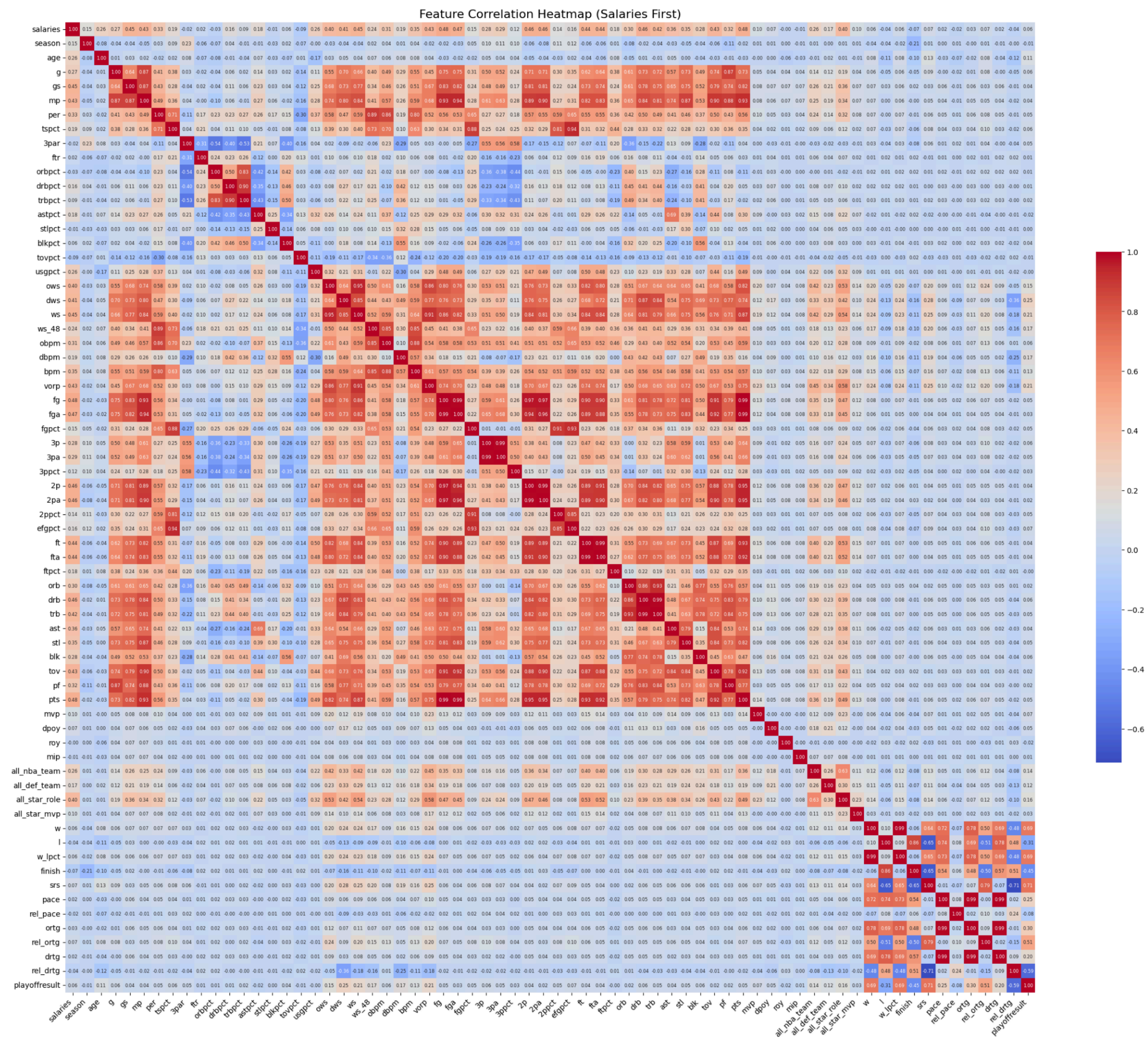
3.1. Correlation Analysis

To understand which features were the most correlated to player salary, I computed the Pearson correlation values and visualized them using two types of heatmaps. One kind of heatmap compared the features with only salary, showing that features like field goals, points, and win shares were highly correlated, relative to other features. Features that were relatively more correlated had values of ~0.5, the highest being field goals at 0.54.



The full feature correlation matrix revealed that many features were highly correlated with one another. A few examples include:

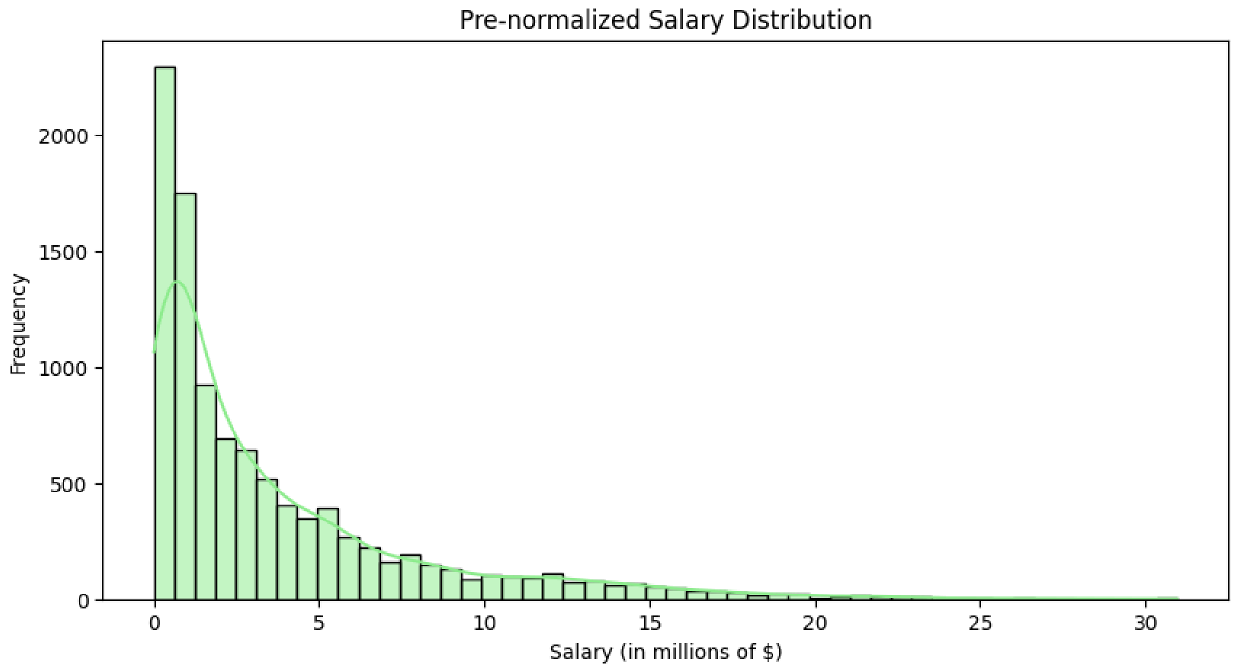
- Field Goals (fg), Field Goals Attempted (fga), and Points (pts) all having an $r = 0.99$
- Win Shares (ws) and Offensive Win shares having $r \approx 0.95$
- Minutes Played (mp) and Games Started (gs) having $r \approx 0.87$

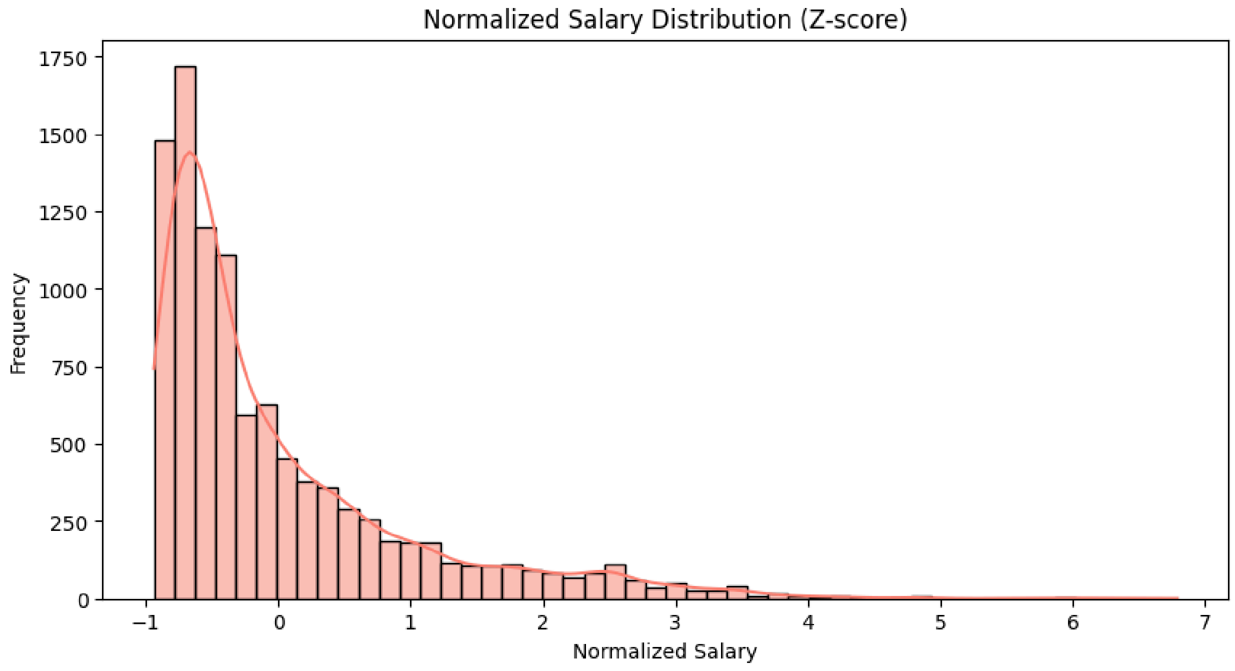


These observations, along with many more suggest that multicollinearity is present within our features. Although this probably won’t affect the performance of tree-based models much, it may affect the stability of coefficients in linear regression.

3.2. Feature Distribution and Label Histograms

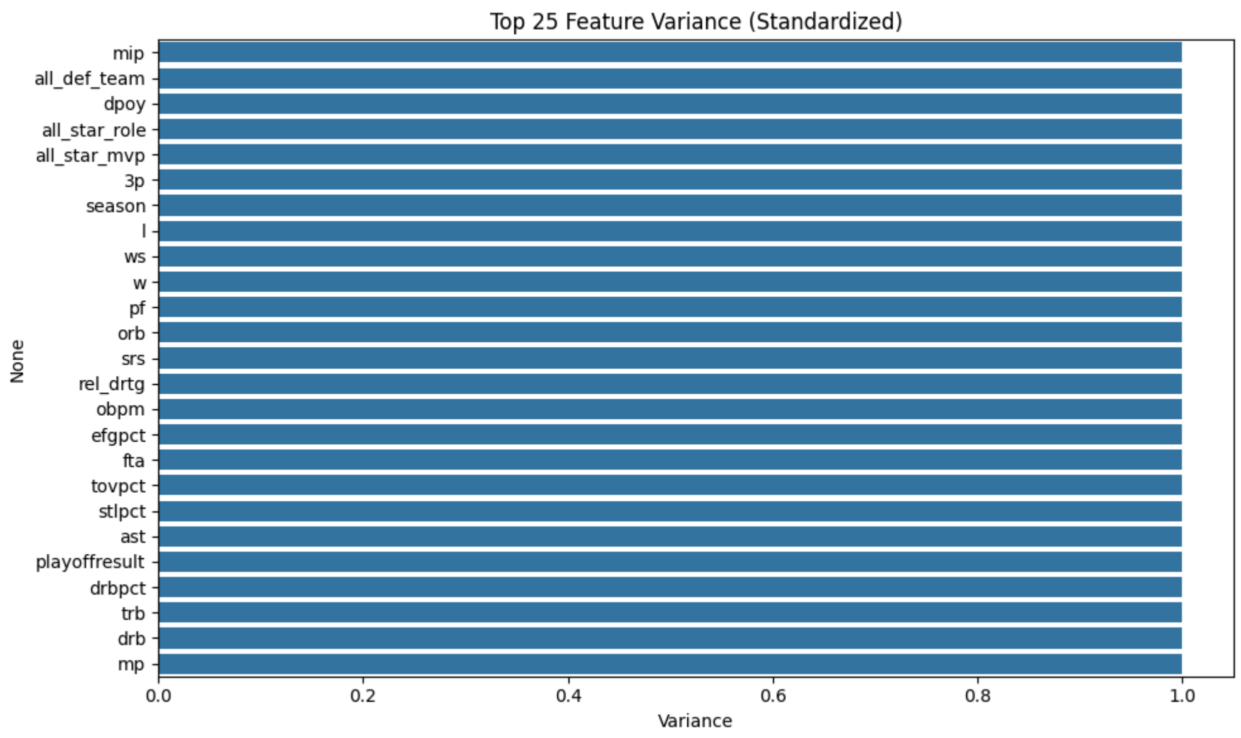
To see the distributions of player salaries, I plotted both the normal and z-scored distributions. Both were heavily right skewed, but the normalized distribution was centered slightly closer to zero, showing the value of the normalization. These histograms show the nature of how salaries are distributed in the NBA, only a few players get those large, max contracts and the difference between the normalized and standard histograms proves that NBA salaries have changed.





3.3. Feature Variance and Standardization

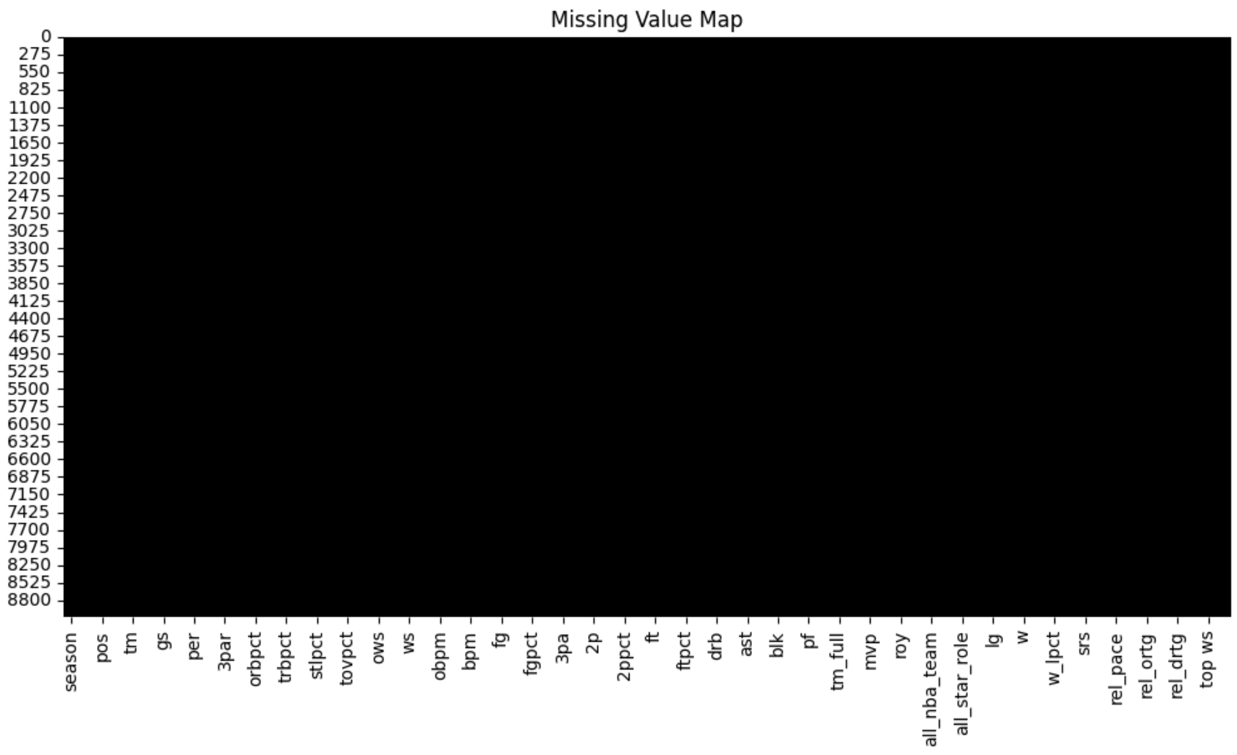
To ensure all features contributed equally to the modeling process, I standardized many of them using z-score. I plotted the top 25 features with the highest standardized variance, many showing very high variance across players.



This is a good sign that there is enough diversity between players, especially in the awards, advanced stats, and team metrics to leverage predictive analysis.

3.4. Missing Values and Imputation Strategy

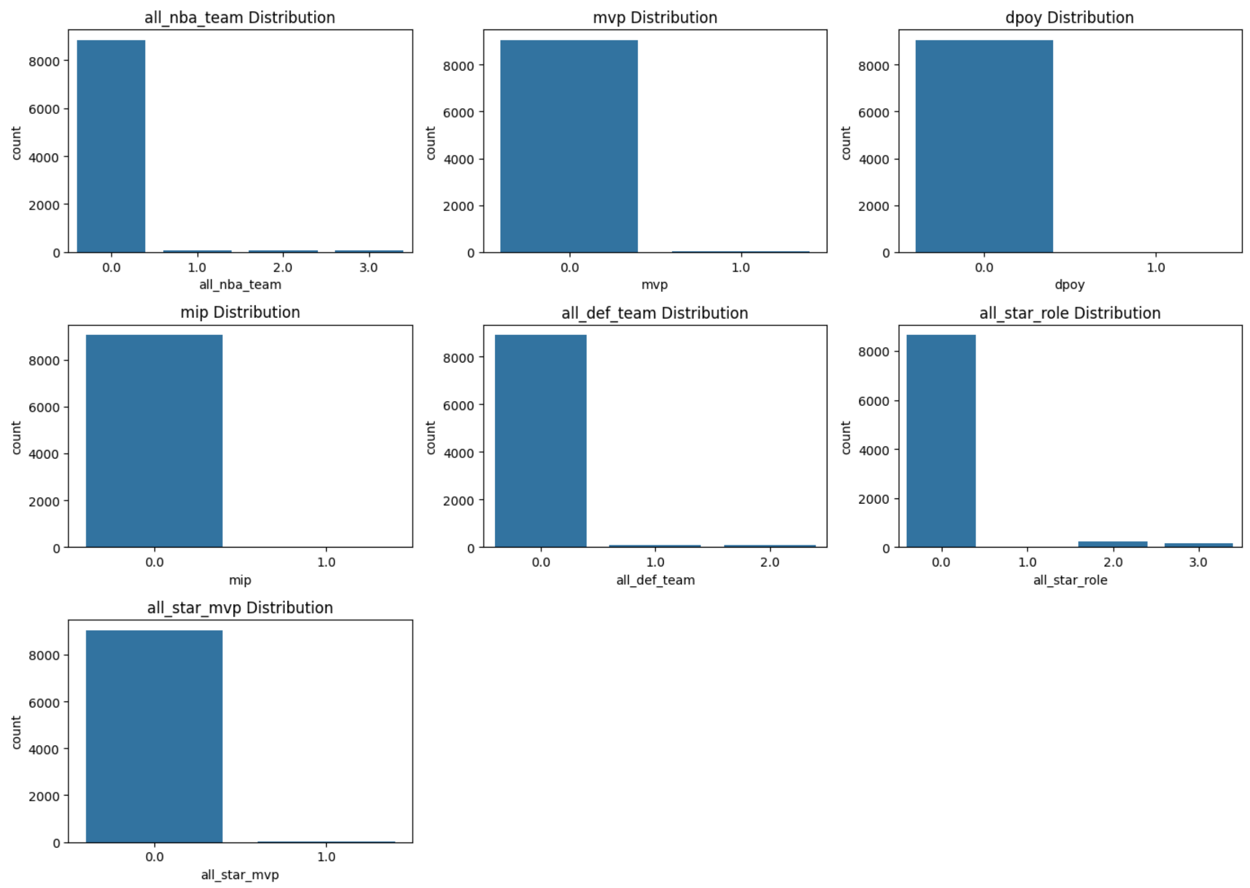
Missing values were primarily dealt with in the data cleaning section. To validate that it worked, I used a heatmap to show the missing values.



The heatmap confirms that I handled the missing values accordingly and they should pose no threat to the remainder of the project.

3.5. Class Imbalance Analysis

Our data didn't have too many categorical features, they came primarily from the award data. There were only 16 MVP winners from over 9000 players, just one example of how small the sample is for award winners.



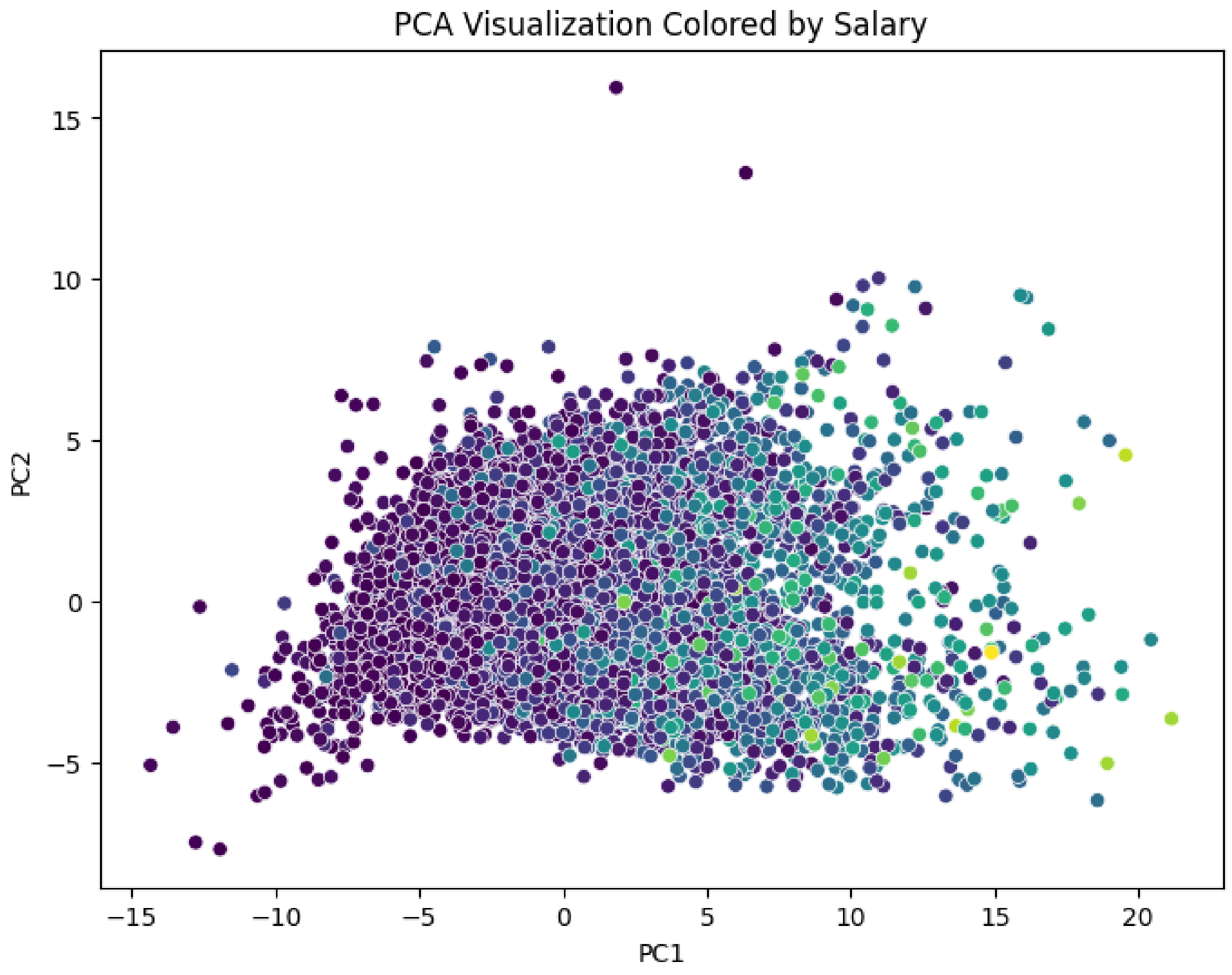
The graph further proves this point, as every major award was completely overwhelmed by 0's.

Despite the heavy imbalance between players who won awards and players who didn't, this isn't a very big issue because this is not a classification problem. However, having mainly zeros in these categories further emphasizes the rarity of these awards without introducing any concern for the project.

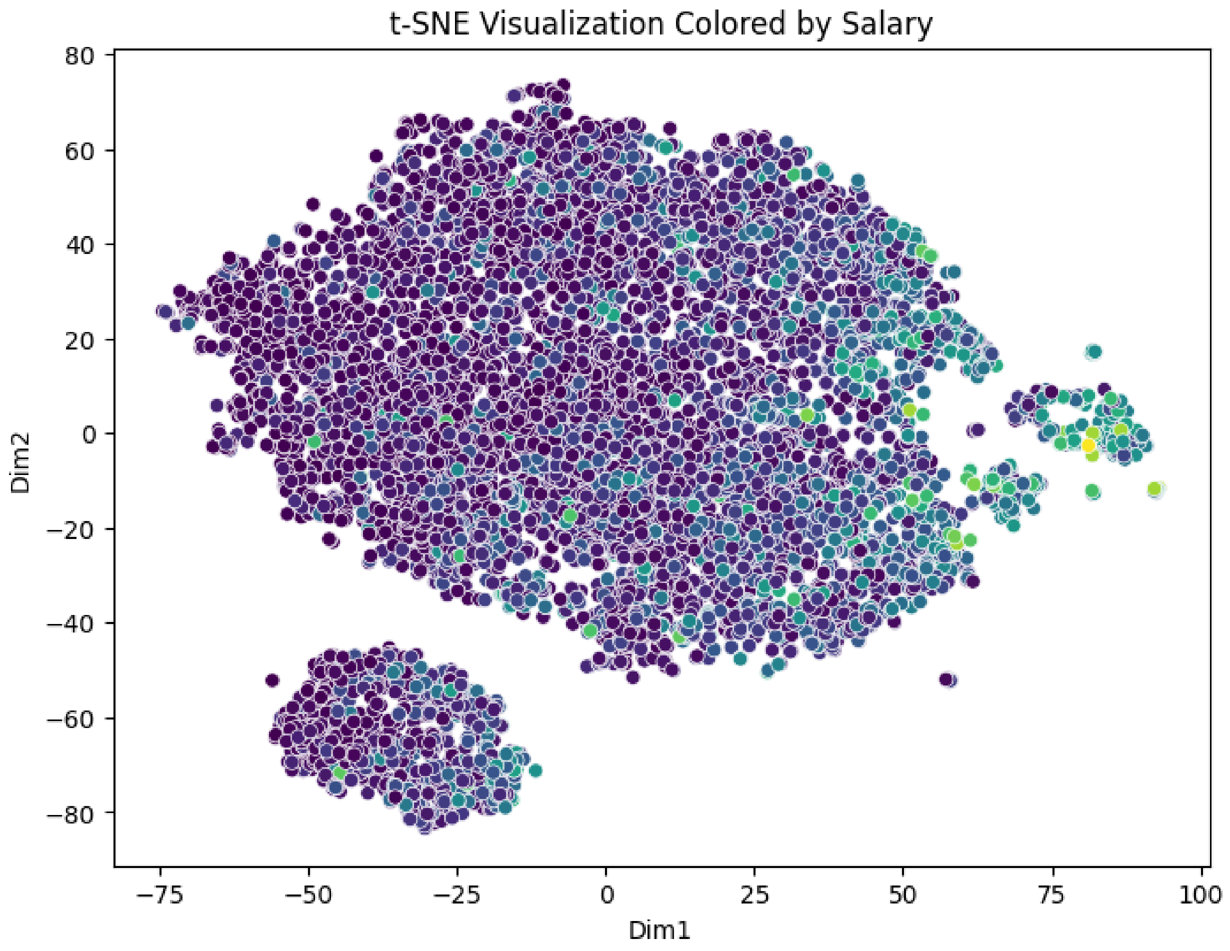
3.6. t-SNE and PCA Visualization

To better evaluate how high earning players separate themselves from the rest of the players, I used two dimensionality reduction techniques, PCA and t-SNE.

The PCA projection shows no clear visual separation of players at different salary levels. The higher paid players tend to trend to the right side of the plot, but the overall spread suggests that simple linear models may not be the best for salary prediction.



The t-SNE echoes a similar sentiment, because the highly paid players are spread out amongst the plot. This means that the distinguishable characteristics of highly paid players are nonlinear and more nuanced.



4. Modeling

4.1. Feature Selection

In order to have each model be as competitive as possible, I had to create a separate feature list for each of the three model types, linear, tree based, and neural networks.

For the linear models, I experimented with VIF testing, as the correlation analysis from the EDA showed that I may have an issue with multicollinearity. I tested three feature sets, the top 12, the top 12 filtered by VIF, and the top 25 featured by VIF, resulting in the following feature sets:

- No VIF: ['pts', 'fg', 'fga', '2p', '2pa', 'ft', 'fta', 'ws', 'drb', 'gs', 'tov', 'vorp']
- Top 12 with VIF: [‘gs’]
- Top 25 with VIF: [‘gs’, ‘all_star_role’, ‘usgpct’]

This confirms that multicollinearity was widespread and problematic.

For the tree based models, the top 15 features were selected based on their feature importance scores. No extra filtering was needed since tree based models handle multicollinearity well. The list of features were: ['gs', 'age', 'usgpct', 'pts', 'season', 'all_star_role', 'per', 'g', 'fg', 'drb', 'drbpct', 'pf', 'dbpm', 'ftpct', 'fga'].

The neural networks feature list was made from the top features of both the linear and tree based models, allowing both linear and non linear patterns to be captured. The list of features were: ['age', '2p', 'all_star_role', 'usgpct', 'gs', 'vorp', '2pa', 'drbpct', 'fga', 'ws', 'per', 'dbpm', 'season', 'g', 'drb', 'pts', 'fg', 'fta', 'tov', 'ftpct', 'ft', 'pf'].

4.2. Models Evaluated

Six models were evaluated using 5 fold cross validation, and RMSE was used as the scoring metric:

- Linear Regression with the three feature subsets
    - Top 12, no VIF: 3,733,023.30
    - Top 12, VIF: 3,924,878.52
    - Top 25, VIF: 3,722,310.07
- Support Vector Regression: ~4,701,500
    - Note: SVR values were similar for all three linear feature subsets.
- Decision Tree Regressor: 4,136,768,67
- Random Forest Regressor: 2,916,948.64
- XGBoost Regressor: 2,874,798.05
- Neural Networks: 3,092,975.80

The best performing models were the XGBoost and Random Forests. The linear models saw some variance depending on the feature set, all of which outperformed SVR. The neural networks had moderate results but showed signs of non-convergence.

4.3. Hyperparameter Tuning

After evaluating all the models, I selected XGBoost and Random Forests since they were the best, along with the best 2 subsets of linear models as a baseline metric. For XGBoost and

Random Forests, I used “RandomizedSearchCV” to tune five key hyperparameters: number of estimators, maximum tree depth, learning rate, subsample ratio, and column subsample ratio for XGBoost, and number of estimators, tree depth, minimum samples per split, minimum samples per leaf, and max features for Random Forests. For Ridge and Lasso regression, a small grid search over regularization strength alpha was performed. I also obtained the CV RMSE for each of these with the tuned parameters. Here were the results:

- XGBoosting: number of estimators = 370, max depth =6 , learning rate = 0.024, subsample ratio = 0.78, column subsample ratio = 0.76, Best CV RMSE = 2,840,093.31
- Random Forest: number of estimators = 274, max depth = 20, minimum samples per split = 7, minimum samples per leaf = 2, max features = sqrt, Best CV RMSE = 2,913,176.87
- Ridge Regression: alpha = 100, Best CV RMSE = 3,721,709.74
- Lasso Regression: alpha = 1, Best CV RMSE = 3,722,310.06

These results reinforce what I discovered earlier, that XGBoost and Random Forests perform the best, significantly better than the linear models.

4.4. Final Model Tuning

After finding the best hyperparameters for the relevant models, I retrained them on the full dataset. This ensures that the final models are trained with the most information possible, without excluding any data for cross validation. The trained models are used for the final evaluation.

5. Results

5.1. Final Model Performance

After training the models on the full datasets with the best parameters, I evaluated their performance with three key metrics:

- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)
- R^2 score

The table below will summarize the results:

Model	RMSE	MAE	R^2
XGBoost	1948698.75	1388979.57	0.809
Random Forest	1666802.12	1124097.28	0.860
Ridge	3696779.47	2695689.94	0.313
Lasso	3696733.45	2693996.53	0.313

Note: Since randomized components are used in model training and cross validation, performance metrics throughout the report may vary slightly.

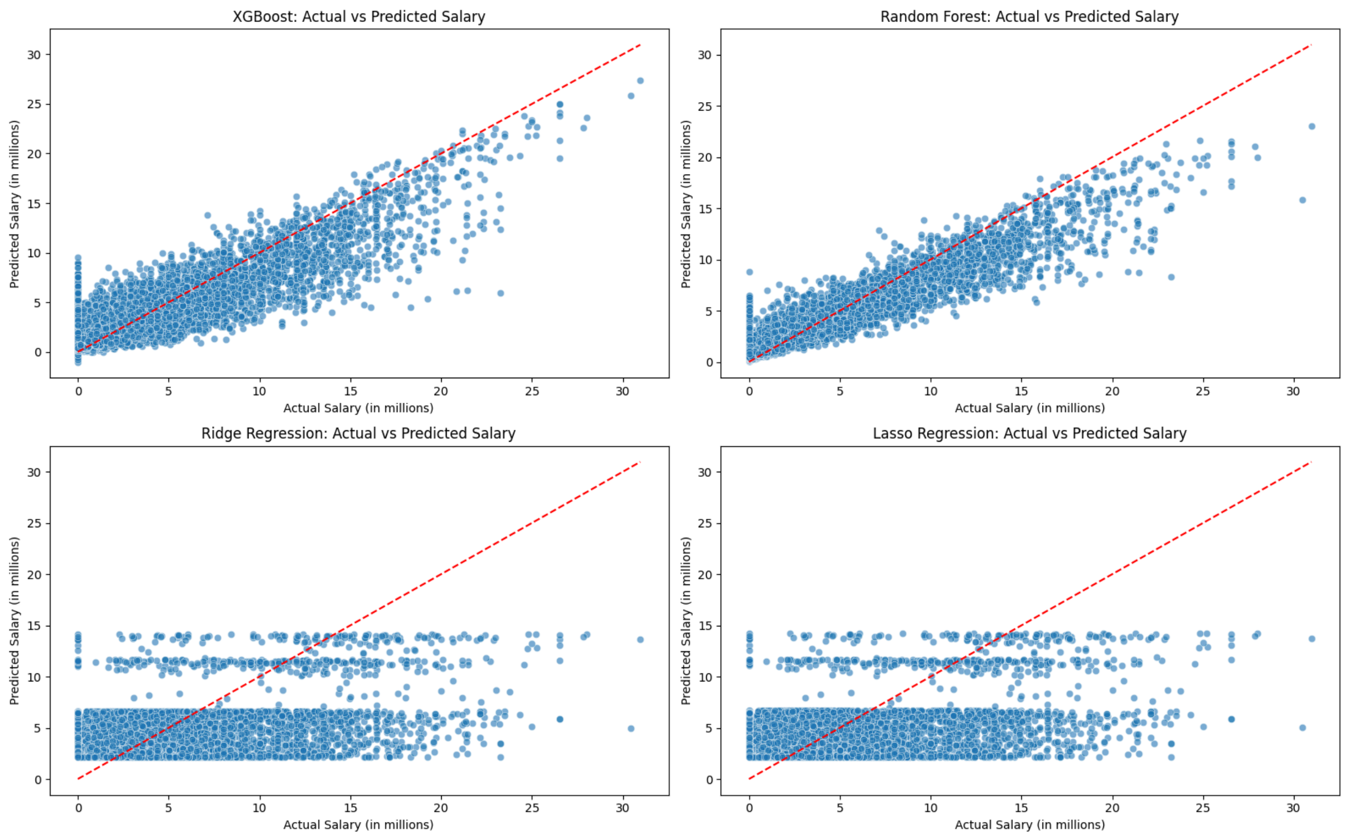
5.2. Model Comparison and Interpretation

After analyzing the results of our tuned models, it's even more clear that Random Forests and XGBoost are the best performing models, with their low errors and high R^2. In contrast, both Ridge and Lasso regression models have high errors and low R^2. These gaps in performance were somewhat expected, after seeing trends in different sections of the EDA such as the PCA, t-SNE, and feature correlation matrix.

6. Conclusions, Limitations, and Future Work

6.1. Conclusions and Key Takeaways

The goal of this project was to build a model that could accurately predict NBA player salaries based on personal and team stats, as well as award related data. After conducting an extensive exploratory data analysis, followed by testing,tuning, and training multiple different models, I have clear results showing that tree based models, particularly XGBoost and Random Forests significantly outperform all other models. Ridge and Lasso regression models, while being useful baselines, were limited in their abilities to capture the salary, probably because of the multicollinearity.



One key takeaway is that the models tend to make more errors with high earning players. I think this is okay, and better than making errors with the majority of players because high earning players generally stick out to general managers anyways, for more reasons than what can be measured by statistics. In the real world, most people can identify which players should earn the top salaries without the use of stats or a model.

6.2. Limitations

Even though I was able to get good results, there were a few limitations worth mentioning:

- Label Skewness: Salary Data was heavily skewed, even after normalization, as there are far less high earners in the league.
- Salary Cap, Contract length/information: It was very hard to find data regarding each franchise's salary cap situations. It was also difficult to find information on the length of a

given contract. Major changes in salary often occur when a player's contract expires and they sign a new one.

- Features: Some features weren't included because they are hard to capture. Things like how marketable a player is, how injury prone they are, and more are hard to quantify.

6.3. Future Work

Looking forward, this project has plenty of room to grow:

- Including more data, like a franchises' financial situation, play-by-play data, or contract details can provide more detail or different insights to what is currently in this project.
- External features like jersey sales, ticket sales, or all star fan votes could capture data regarding a players popularity, something that isn't very tangible with the current data.
- Analytical dashboarding can provide a way to share this data, making it more applicable and usable in the real world.
- I can broaden this project to outside the NBA, including different sports, or different basketball leagues like the G-League, international leagues, or the WNBA.
- Although player info is included in the project, I didn't explore it nearly as much as the other types of data I collected, going back to that could provide another unique insight.

7. Appendix

This link will take you to a github repository which will have all the csv's, the code, the presentation, as well as the sources for where I got my data.

https://github.com/zach200311/NBA-Salary-Prediction-Project