Predicting Basketball Salaries

CSP 571: Data Preparation and Analysis Final Project

Aleksei Sorokin. asorokin@hawk.iit.edu
Arjuna Anilkumar. aanilkumar@hawk.iit.edu
Hardev Ranglani. hranglani@hawk.iit.edu
Hyungtaeg Oh. hoh12@hawk.iit.edu

Table of contents

Overview

The objective of this project is to identify undervalued and overvalued NBA players. To do this, we will work to predict professional basketball player salaries based on raw and aggregated statistics about their performance and abilities. Our team will clean data to improve modeling efficiency and performance. Building models include constructing models such as multiple linear regression, decision trees, and random forest, each with tunable parameters that can be refined to improve model metrics. These models are evaluated in terms of player salary and then related back to identifying undervalued and overvalued players

Data Collection and Linking

Our primary data source contains player stats and salaries for each year between 1950 and 2017. The dataset contains aggregate individual stats from basic box-score attributes such as points, assists, rebounds, etc. to more advanced money-ball like features such as Value Over Replacement. This dataset was directly downloaded from Kaggle **[2]** as a .xlsx file which could easily be read into R. One difficulty of this dataset is handling players who have been traded and thus have multiple records for the same year. We decided to aggregate the totals across all teams for traded players and drop individual-team statistics. This required adding the new team category 'TOT': total aggregated yearly statistics for a traded player.

Primary dataset attributes

year, name, salary, position, age, team, games, games started, minutes played, player efficiency ranking, true shooting %, 3PA over FA, free throw rate, offensive rebound %, defensive rebound %, 'total rebound %, assist to turnover %, steal %, block %, turnover %, usage %, offensive wins shares, defensive wins shares, win shares, win shares over 48, offensive box plus-minus, defensive box plus-minus, box plus-minus, value over replacement player, field goal, field goal attempts, field goal %, 3 pointer, 3 pointer attempts, 3 point %, 2 pointer, 2 pointer attempts, 2 point %, effective field goal %, free throws, free throw attempts, free throw %, offensive rebounds, defensive rebounds, total rebounds, assists, steals, blocks, turnovers, personal fouls, points.

Our secondary data source contains player ratings from the NBA 2K video game series for each year between 2016 and 2020. This dataset was sourced from the My Team DataBase (MTDB) **[3]** using web scraping. Each year's data took the form of multiple web-pages with tables of player data. By specifying the number of pages and using URL patterns, we could extract the tables from each page for each year. When all pages for all years were scraped, tables from the same year were concatenated to produce the raw secondary dataset. One difficulty of this dataset is that multiple versions of a single-player have been put into the game. For example,

there are two versions of LeBron James in NBA 2K16: one for regular season and one for the postseason. This was addressed by taking the player version with a higher overall rating. When multiple versions of a player share the same overall ranking, the player is chosen with the greater outside rating, then inside rating, then greater playmaking, … We chose this approach as teams are more likely to pay for the player's max potential.

Secondary dataset rating attributes

name, position, overall, outside, inside, playmaking, athleticism, defending, rebounding

In order to link the primary and secondary dataset, we treated the player name as the record unique ID. Names were first set to all lower case, and then misspellings were checked using Levenshtein distance. Names with small Levenshtein distance were manually checked for misspellings. This task proved non-trivial due to many different players who have similar names. For example, Zoran Dragic and Goran Dragic are siblings with very similar names who both played in the NBA. Overall, around 10 names were identified as being misspelled in either the primary or secondary dataset. A major limitation of our datasets is the only overlapping years of data are 2016 and 2017. While both the primary and secondary datasets alone contain large amounts of player data, the overlap of two datasets contains only around 750 records. As further discussed in our future work, acquiring more data would allow us to build better models with greater prediction power. Overall, our combined dataset (primary + secondary datasets linked) contains at most 2 versions of any given player: 1 for 2016 and 1 for 2017.

Data Preparation and analysis

Data cleaning

The data obtained had to be cleaned a little bit. The column names were changed to their appropriate abbreviations. There were players with numbers in their names that signified that the player was a legendary version. Such players were removed from the dataset. Any dynamic versions of the players were also removed from the dataset. In the secondary dataset, there were multiple versions of the same player so all duplicates were removed. The Levenshtein distance algorithm was used to do spell checking in the names variable and replaced any misspelled names with correct names.

Missing values

To detect any missing values a combination of methods was used. The 'sapply' function was also used to detect missing values. The output gave a sum of missing values for each variable in the primary dataset. The 3P% variable had the most missing values with 1562 values missing out of a total of 1998 missing values in the dataset. Missing values were replaced by the

mean of their respective columns as represented by the figures below. The code was then run again to check for any other missing values in the dataset

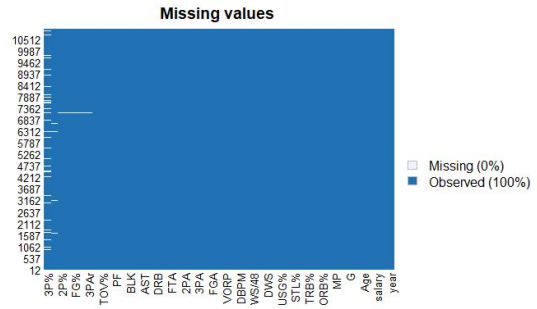
year	name_p	salary	Pos	Age	Tm	G	GS	MP	PER	TS%	3PAr	FTr
0	0	0	0	0	0	0	0	0	0	24	30	30
ORB%	DRB%	TRB%	AST%	STL%	BLK%	TOV%	USG%	OWS	DWS	WS	WS/48	OBPM
0	0	0	0	0	0	19	0	0	0	0	0	0
DBPM	BPM	VORP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%	eFG%
0	0	0	0	0	30	0	0	1562	0	0	40	30
FT	FTA	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS	
0	0	233	0	0	0	0	0	0	0	0	0	

Figure 1: Missing value count.

```
df_primary$`3P%`[is.na(df_primary$`3P%`)] <- mean(df_primary$`3P%` , na.rm = T)
df_primary$`FT%`[is.na(df_primary$`FT%`)] <- mean(df_primary$`FT%` , na.rm = T)
df_primary$`TOV%`[is.na(df_primary$`TOV%`)] <- mean(df_primary$`TOV%` , na.rm = T)
df_primary$`FG%`[is.na(df_primary$`FG%`)] <- mean(df_primary$`FG%` , na.rm = T)
df_primary$`2P%`[is.na(df_primary$`2P%`)] <- mean(df_primary$`2P%` , na.rm = T)
df_primary$`eFG%`[is.na(df_primary$`eFG%`)] <- mean(df_primary$`eFG%` , na.rm = T)
df_primary$`TS%`[is.na(df_primary$`TS%`)] <- mean(df_primary$`TS%` , na.rm = T)
df_primary$`3PAr`[is.na(df_primary$`3PAr`)] <- mean(df_primary$`3PAr` , na.rm = T)
df_primary$`FTr`[is.na(df_primary$`FTr`)] <- mean(df_primary$`FTr` , na.rm = T)
```

Figure 2: Replacing missing values with average values.

A map of missing value was plotted using the 'missmap' function in r using the Amelia Library. This provides a great way to visualize the number of missing values in the dataset. The figures below show the missmap plot before and after treating missing values.



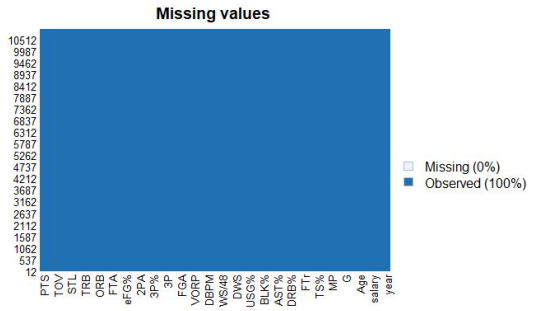


Figure 3: Missing value before cleaning

Figure 4: Missing value after cleaning

EDA

In this section, we explored the primary dataset and the salary variable which is our target variable. Below in figure 5 is a histogram and a boxplot of the salary variable and we can see that the distribution for the salary variable is positively skewed as is the case for all salary data. There are also a few outliers in our data that are dealt with in the outliers section of the report.

Some stats for the salary variable are 7.8 million average salary for all players in the data set, the median is 5.2 million and the most common salary for the players in this dataset is 1.3

million. The distribution as seen in the histogram is positively skewed as is the case for all salary data.

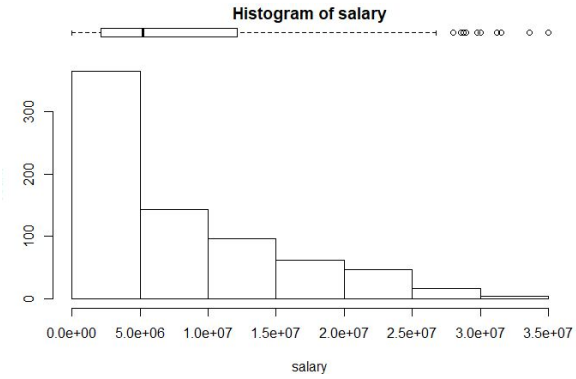


Figure 5: Histogram of salary

Figures 6 and 7 show the number of players in each position and the average salary for each position. Based on the figures the small forwards position has the highest average salary and while containing the least number of records.

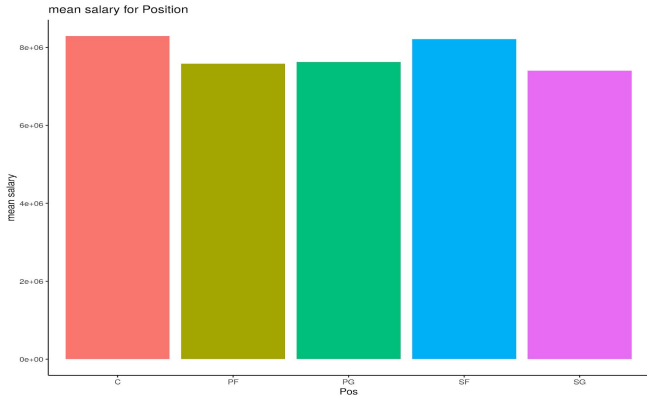


Figure 6: Mean salary for each position

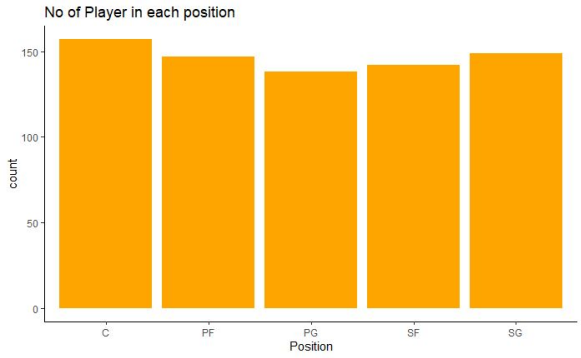


Figure 7: No of players in each position

Figure 8 is a plot that shows more in depth plots of the number of players in each team grouped by their positions. These plots give us valuable information about the composition of player positions in each team. Figure 9 shows the mean salary per team. From these plots, we can see that the Cleveland Cavaliers (CLE) has the highest average salary but also has a smaller number of players in the dataset.

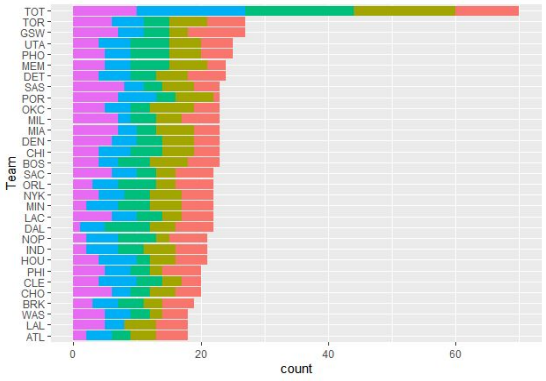


Figure 8: No of players in each team with position

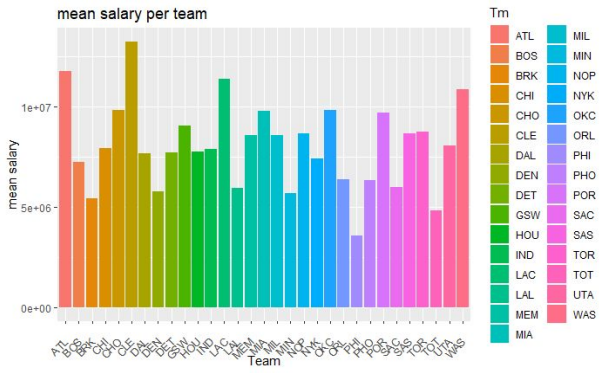


Figure 9: Avg salary per team

Correlations

The next plot in figure 10 shows the complete density plots for all the variables in the primary variables. This gives us some valuable insight about the variables in the dataset and how some variables are explaining the same thing. For example, take 2 points (2P) and 2 point attempts (2PA) variables; they have the same distribution and suggest that they are correlated and don't give any new information.

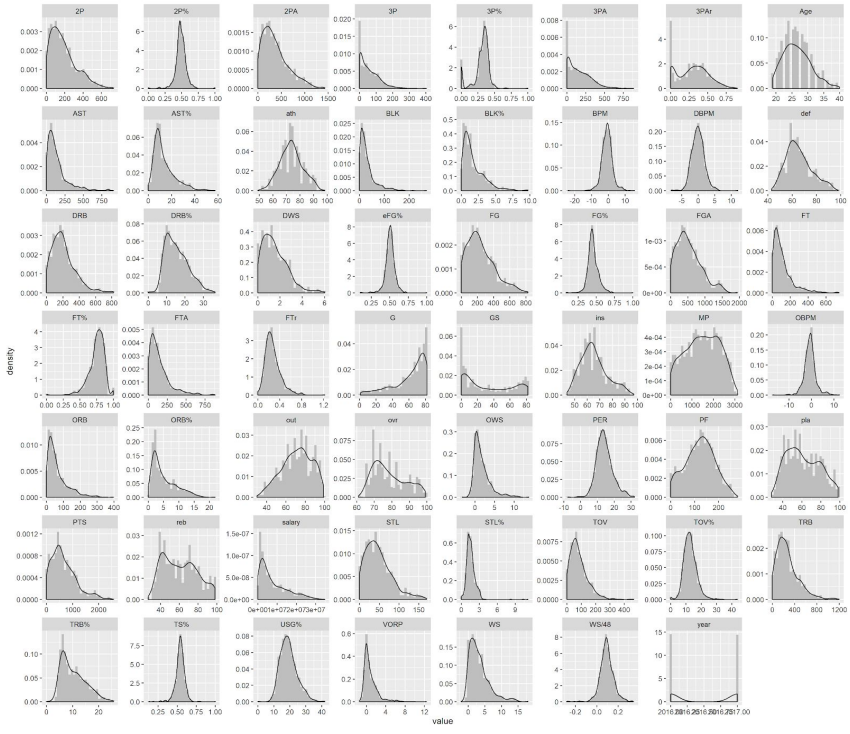


Figure 10: Density plot of all numeric variables

Using the correlation plot in figure 11 and the density plot of all variables in figure 10, we can see that 2P and 2PA are highly correlated along with some other variables. Thus the 2PA

variable was removed from the dataset. The following variables were also removed from the primary and complete datasets. "G", "3PAr", "FTr", "ORB%", "DRB%", "OWS", "DWS", "WS/48", "OBPM", "DBPM", "FGA", "3PA", "2PA" "FTA", "ORB", "DRB", "TRB", "STL", "BLK" and "TOV". This reduced the number of variables from 51 to 38 in the dataset. Figure 12 shows the correlation plot for the relevant 38 variables and it can be seen that there is no strong correlation between the variables. Figure 13 shows the correlation between variables for the complete dataset after removing unwanted variables from the dataset.

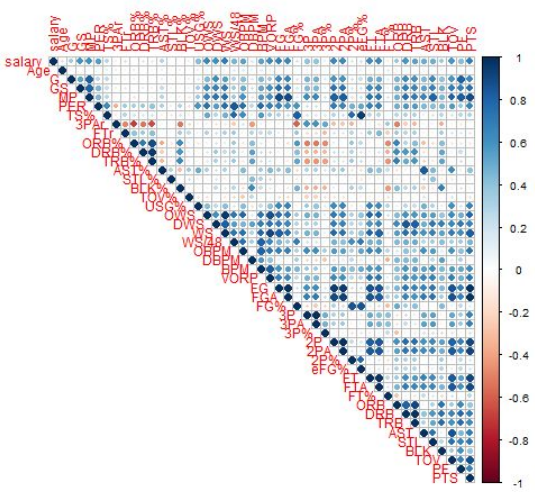


Figure 11: Plot before removing variables

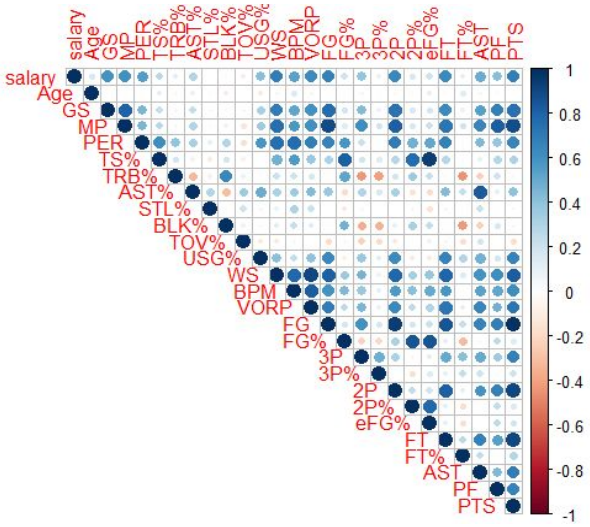


Figure 12: Plot after removing variables

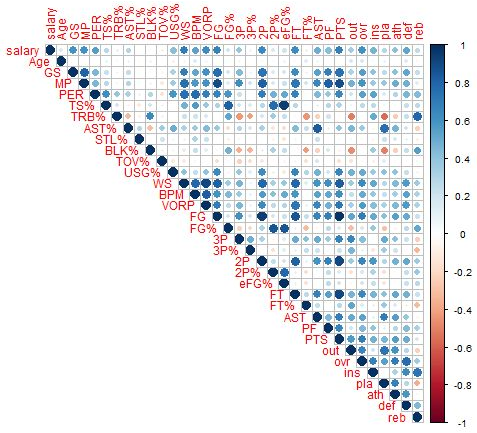


Figure 13: Correlation plot for the complete dataset

Outliers

To detect outliers in the dataset, a multivariate approach using Cook's distance was used. Cook's distance is a measure computed with respect to a given regression model and therefore is impacted only by the variables included in the model. It computes the influence exerted by each data point (row) on the predicted outcome. The Cook's distance for each observation *i* measures

the change in fitted Y for all observations with and without the presence of observation *i*, so we know how much the observation *i* impacted the fitted values. In general, observations that have a cook's distance greater than 4 times the mean may be classified as influential. Since the number of observations is low in our dataset a cook's distance of 9 times the mean was used in the model. Figure 14 gives a good visual representation of the outliers in our dataset.

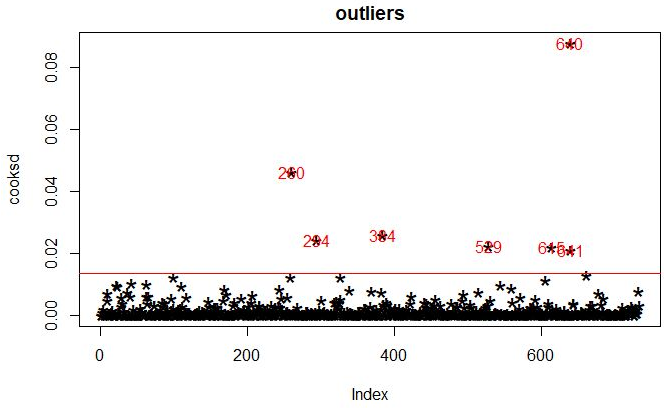


Figure 14: Outlier plot

The list of players who are classified as outliers is given in figure 15. Upon closer inspection, this is a list of really good players who have impressive basketball stats or some very bad stats for that particular year. There are players who have very low minutes played which suggests that they may be starting their careers. For example, Stephen Zimmerman has 108 minutes played and very low points and games started variables which suggests that he just started his career as a basketball player this year. With more research on these players, it was established that they are considered outliers because these stats are from the year 2016-2017 which is a very narrow range. So for this reason and because our dataset is low on observations already, we decided to include these players in our dataset.

	name	year	salary	Pos	Age	Tm	GS	MP	AS...	WS	FG%	3P%	2P%	FT%	PF	PTS
	<chr>	<fctr>	<dbl>	<fctr>	<dbl>	<fctr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
260	isaiah thomas	2017	6261395	PG	27	BOS	76	2569	32.6	12.6	0.463	0.3790000	0.528	0.909	167	2199
294	jarrett jack	2017	2328652	PG	33	NOP	0	33	20.3	0.0	0.667	0.0000000	1.000	1.000	4	6
384	karl anthony towns	2017	6216840	C	21	MIN	82	3030	13.2	12.7	0.542	0.3670000	0.582	0.832	241	2061
530	nikola mirotic	2016	5782450	PF	24	CHI	38	1646	9.4	3.9	0.407	0.3900000	0.430	0.807	151	777
616	sam dekker	2017	1794600	SF	22	HOU	2	1419	7.7	3.1	0.473	0.3210000	0.591	0.559	83	504
641	stephen curry	2017	34682550	PG	28	GSW	79	2638	31.1	12.6	0.468	0.4110000	0.537	0.898	183	1999
642	stephen zimmerman	2017	1312611	C	20	ORL	0	108	5.3	0.0	0.323	0.2711419	0.323	0.600	17	23

Figure 15: List of outliers

Feature Selection

Since there are 38 variables in the data, feature selection was used to find the optimal subset of variables for both the primary dataset and the complete dataset. Automated F-Test-Based Backward Selection was used to get an optimal subset of variables.

Selected primary dataset predictors:
year, Pos, Age, Tm, GS, TS%, AST%, WS, VORP, FG%, 3P, FT, PF
Selected complete dataset predictors:
Age, GS, MP, USG%, WS, PF, out, ovr

Variance inflation factors (VIF) were used to detect multicollinearity among the selected variables. Variance inflation factors (VIF) measures how much the variance of the estimated regression coefficients are inflated as compared to when the predictor variables are not linearly related. Figures 16 and 17 show the list of selected variables and their VIF score. From this, we can see that the selected subset of variables has very low VIF values with a maximum of 15 on the primary dataset predictors and a maximum of 8 on the complete dataset predictors. From this, we can conclude that there is little to no multicollinearity among the selected predictor variables.

WS	`FG%`	VORP	`TS%`	TmTOT	PosPG	FT	`AST%`	`3P`	PosSG
14.914773	9.299794	8.728394	7.144946	5.086995	4.190961	4.167082	3.650485	3.456991	3.092757
TmTOR	GS	TmPHO	TmGSW	TmUTA	TmMEM	TmCHI	TmSAS	TmLAC	TmOKC
3.037526	2.863495	2.858784	2.812127	2.793899	2.720141	2.692851	2.678805	2.637327	2.577475
TmMIN	TmBOS	TmDET	TmNYK	TmPOR	PF	PosSF	TmORL	TmCHO	TmHOU
2.573613	2.557200	2.547414	2.535575	2.526887	2.499607	2.498992	2.488734	2.472843	2.410641
TmCLE	TmIND	TmPHI	TmDEN	TmNOP	TmMIL	TmDAL	TmSAC	TmMIA	TmLAL
2.398189	2.392428	2.383919	2.327086	2.320728	2.256420	2.255879	2.252197	2.238935	2.203021
TmBRK	TmWAS	PosPF	Age	year2017					
2.172056	2.134109	1.908324	1.280955	1.049395					

Figure 16: List of the selected variable for the primary dataset

MP	PF	GS	WS	ovr	out	`USG%`	Age
7.984500	3.929993	3.197309	3.078852	2.832498	2.563631	1.613312	1.136827

Figure 17: List of selected variables for the secondary dataset

Train-Test Split

Stratified sampling was used to split the data into train and test data for both the primary and complete datasets. The CreateDataPartition function from the caret package in r was used **[4]**. After splitting the data, the train data had 590 observations and the test data had 144 observations.

Modeling

Linear Regression

The first models we built were linear regression models. This includes simple, ridge, lasso, and elastic net linear regression models. Models were trained on both the primary and combined datasets in order to understand the difference in predictive power from our secondary datasets.

For simple linear regression a total of four models were trained from every combination of all variables/optimal subset of variables and primary/combined (primary+secondary) datasets. Specifically, the models were constructed from data on the primary dataset with all variables, combined dataset with all variables, primary dataset with an optimal subset of variables, and combined dataset with an optimal subset of variables. The two models trained on the complete set of variables had higher test R^2 values and lower test MSE values when compared to the two models trained on the optimal subset of variables. Models trained on the primary dataset had lower test R^2 values and higher test MSE values when compared to models trained on the combined dataset. Overall, of the four standard linear regression models, the model trained on the combined dataset with all possible predictors performed the best, yielding a test R^2 of .544 and a test RMSE of around $4.82M.

Ridge, lasso, and elastic net models were all trained in a similar manner. The glmnet R package **[5]** was utilized to train elastic net models using cross-validation. glmnet generalizes the standard penalty function for elastic net models with parameters λ and α.

Standard Penalty Function: $\| y-XB \|^2 + \lambda_1 \| B \|^2 + \lambda_2 \| B \|_1$

glmnet Gaussian Objective Function: $\| y-XB \|^2/(2n_{obs})+\lambda[(1-\alpha) \| \beta \|^2/2 + \alpha \| \beta \|_1]$

Each penalty function can be thought of as a combination of the ridge and lasso penalty functions. When α=1 the model is trained by lasso regression. When α=0 the model is trained by ridge regression. To optimally select α and λ values we trained elastic net models with α taking on every value between 0 and 1 inclusive with a step size of 0.05 ~ seq(0,1,by=.05). Cross-validation built into the glmnet package allows for λ to be optimally chosen for each α value. This process was repeated for both the primary and combined datasets (a total of 42 models), again to see the effect of our secondary dataset in increasing or decreasing predictive power. It should be noted that all possible variables were used to train the models as elastic net models automatically perform variable selection, so there was no need to use our optimal subset of variables. Unsurprisingly, models built with the primary dataset performed worse on the test set than models built with the combined dataset. Overall, the optimal α value for models trained on our primary dataset was 0 (ridge regression) with a corresponding optimal λ value of around 52k. The optimal α value for models trained on the combined dataset was also 0 (ridge regression) with a corresponding optimal λ value of around 50k. The R^2 for the best model

(trained on a combined dataset with α=0 and $\lambda\sim$=50k) attained a test R^2 of 0.62 and a test RMSE of around $4.4M.

Decision Tree

For Decision Tree, we used the package 'DecisionTreeRegressor' from the sklearn library. The model was built on the combined dataset to evaluate model performance. However, the model performance was not really good and the R2 metric on the test data was only 38% compared to 64% on the train data. Also, the feature importance plot returned only a few variables as important. Hence, it was evident that the Decision tree model was not performing well.

Random Forest

For the Random Forest model, we used the package 'RandomForestRegressor' from the sklearn library. We also tried the grid-search hyperparameter tuning to get the best hyperparameters and the "validation_curve" package in sklearn to get the results of individual tuning of each hyperparameter. However, even after getting the best hyperparameters, the model performance did not improve much on the test data. We obtained an R2 of 49% on the test data compared to a R2 of 81% on train data.

Gradient Boosted Trees

For the Gradient Boosting Trees(or Gradient Boosting Machines, GBM) model, we used the package 'GradientBoostingRegressor' from the sklearn library. We also tried the grid-search hyperparameter tuning to get the best hyperparameters and the "validation_curve" package in sklearn to get the results of individual tuning of each hyperparameter. However, even after getting the best hyperparameters, the model performance did not improve much on the test data. We obtained an R2 of 48% on the test data compared to a R2 of 94% on train data

Optimal Model

Of the models discussed in this section, the ridge regression (elastic net with α=0) model trained on the combined dataset with all predictors performed best on the test set with the highest test R^2 and lowest test MSE. Therefore this model was chosen to help identify underrated and overrated players and was chosen to be deployed, as further discussed in later sections of the paper.

Performance Considerations

While we worked to train and test many different models with optimized hyperparameters, our overall model performance was less than impressive. With our optimal model achieving just over a 0.6 R^2 on the test set and RMSE of over $4M, it is difficult to trust predictions from our model. This is to say, the following sections on identifying underrated and overrated players and using our deployed model should be taken with a grain of salt.

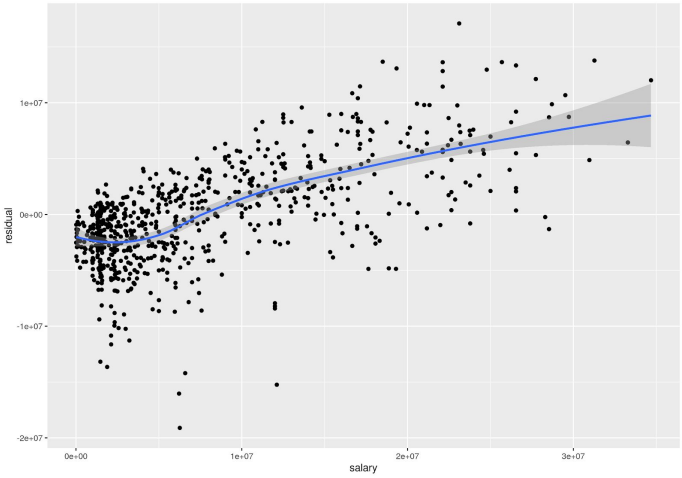


Figure showing residual vs salary from optimal model salary predictions. The optimal model tends to overestimate players with lower salaries and underestimate players with higher salaries.

There are a few possible explanations for models' weak performance. First, the models were trained with small datasets. With under 1,000 records, many models were not able to be adequately trained. Moreover, using cross-validation on such a small dataset often led to overfitting by the time parameters had converged. Second, the outliers, while not data entry errors, cause many models to arrive at highly skewed parameters. One idea was to remove outliers in order for our models to be trained on a more robust/uniform dataset. However, this approach further limits the scope of our problem and further reduces the already small size of our dataset. Therefore, we keep outliers in the dataset but recognize it may hurt model performance. Third, our model is being trained to predict figures settled on by NBA higher-ups who take into consideration salary caps, trades, and other team financials. Not every team can afford to pay a player their statistical worth, and therefore players often end up with salaries that vary greatly from what they should earn based on their performance.

Underrated and Overrated Players

The goal of this work is to identify underrated and overrated players by salary. To do so, we looked at a player’s actual 2016 salary versus the optimal model’s predicted 2016 salary.

An underrated player was defined to be any player who our model predicts to be making at least twice as much as they make in reality. Using this criterion our model identified 92 underrated players in 2016. To evaluate the validity of these identifications, we looked at those 92 player's real 2017 salaries. Around 62% of the 92 underrated players in 2016 saw over a 25% increase in salary for 2017. This is an encouraging metric for players identified as underrated who could use such a model to argue for a higher salary.

An overrated player was defined to be any player who our model predicts to be making no more than half as much as they make in reality. Using this criterion our model identified 31 overrated players in 2016. Again, we compared these overrated players' 2016 salaries to their 2017 salaries. However, this time we saw almost the opposite effect. Instead of overrated player's salaries decreasing, around 69% of the identified overrated players actually saw an increase in salary from 2016 to 2017. This is relatively unsurprising for a few main reasons. One, our model tends to underestimate players who have large salaries. Therefore, it is likely that players with high salaries earn their pay, but our model may still be identifying them as overrated. Thus, when their salary increases, we are surprised from a modeling perspective, but teams are unsurprised based on real performance. Two, it is difficult to justify cutting a player's salary. Perhaps they were injured, had bad teammates, or just had an off-year. From a managerial perspective, it may be difficult to justify decreasing a super-star's salary based solely on statistics from the previous season.

Deployment

For the deployment of our optimal model, we used the Plumber R package **[6]**. Plumber provides an easy to use interface for users to interact with an R model. The user does not need to have R or any application other than a terminal, or perhaps a web-browser. When our model is deployed it can be queried directly via the command line or interactively with the Swagger API built by Plumber. Take for example LeBron James in 2016. Assuming our model is deployed, LeBron could open the command line and type

```
curl-X POST
'http://<IP>/predict_salary?Age=31&G=76&GS=76&MP=2709&PER=27.5&TS.=.588&X3PAr
=0.199&FTr=0.347&ORB.=4.7&DRB.=18.8&TRB.=11.8&AST.=36.0&STL.=2.0&BLK.=1.5&
TOV.=13.2&USG.=31.4&OWS=9.6&DWS=4&WS=13.6&WS.48=0.242&OBPM=6.9&DBPM
=2.3&BPM=9.1&VORP=7.6&FG=737&FGA=1416&FG.=0.520&X3P=87&X3PA=282&X3P.
=0.309&X2P=650&X2PA=1134&X2P.=0.573&eFG.=0.551&FT=359&FTA=491&FT.=0.731&
ORB=111&DRB=454&TRB=565&AST=514&STL=104&BLK=49&TOV=249&PF=143&PTS
=1920&out=94&ovr=99&ins=89&pla=91&ath=92&def=91&reb=91'
```

with <IP> replaced by the IP address where the model is currently deployed. The query would return his predicted salary of $26.1M, which would discourage him when compared to his actual $31M salary in 2016. However, I doubt LeBron James would use the command line. It would be much easier to open up the web app which provides a friendly UI that prompts him for various input fields with a clear description of each statistic.

Future Work

Future work on this project should focus on collecting more data related to NBA basketball player's salaries. With less than 1,000 records, many of our models were not able to be trained to their full potential. One possible source of data is ESPN who maintains current and past player data. The main advantage of using ESPN over our current player data would allow the overlap of primary and secondary datasets to include 2016-2020 (all the years the secondary dataset is available). In fact, a script for scraping all ESPN NBA player data can be found at *deprecated/load_data.Rmd* in our GitHub repository **[8].** Another secondary data source to consider would be NBA trades, salary caps, and team financials data. While a player's performance undeniably plays a role in determining their salary, the final figure is much more complex to arrive at. Teams determine a salary based on trades with other teams (which may involve multiple players and waivers), the salary cap (the total amount a team can spend on its players), and various other financial decisions that must be made by higher-ups in an organization when determining a player's final salary. For example, the NBA imposes minimum and maximum player salaries based on the number of years the player has been in the league **[9]**. Including data on such a floor and ceiling along with the number of years a given player has been in the league would help constrain model predictions and hopefully improve performance. Specifically, we expect to see a higher R^2 value due to an increased explanation in the variation among the salaries of players with similar performance statistics.

References

[1] Wesolowski-Mantilla, A. (2017). A Breakdown of NBA Salaries. Northwestern Business Review.
https://northwesternbusinessreview.org/a-breakdown-of-nba-salaries-d8c37c56a4ee

[2] Blanco, F. (2018). NBA - Advanced & Basic Season Stats (1950-2017). Kaggle.
https://www.kaggle.com/whitefero/nba-players-advanced-season-stats-19782016

[3] My Team Database (MTDB). (2015).
http://mtdb.com/20

[4] Max Kuhn (2020). caret: Classification and Regression Training. R package version 6.0-86.
https://CRAN.R-project.org/package=caret

[5] Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22.
http://www.jstatsoft.org/v33/i01/

[6] Trestle Technology, LLC (2018). plumber: An API Generator for R. R package version 0.4.6.
https://CRAN.R-project.org/package=plumber

[7] Ryan Prete. NBA Superstars Face Tax Bills in the Millions. (2018). Bloomberg Tax.
https://news.bloombergtax.com/daily-tax-report-state/nba-superstars-face-tax-bills-in-the-millions

[8] csp471_fp GitHub repository. GitHub.
https://github.com/alegresor/csp571_fp

[9] Kerry Miller. How NBA Free Agency, Salary Cap Work. (2018). Bleacher Report.
https://bleacherreport.com/articles/2787871-how-nba-free-agency-salary-cap-work

[10] Josh Rosson. NBA Salary Predictions using Data Science and Linear Regression. (2019). Towards Data Science.
https://towardsdatascience.com/nba-salary-predictions-4cd09931eb55

[11] Ben Glover. NBA announced record salary cap for 2016-17 after historic climb. (2016). Sports Illustrated.
https://www.si.com/nba/2016/07/03/nba-salary-cap-record-numbers-2016-adam-silver