Predicting NBA Players' 2K21 Ratings

Ben Huston and Eli Standard

University of Vermont

CS/STAT 287

5/11/2022

Summary/abstract:

Our project investigates the question: Can defensive stats predict NBA player rating in the 2k21 video game? Per-game statistics from the 2020-2021 season serve as predictor variables for multiple linear regression models that are used to predict player overall scores (OVRs). We hope that we can draw conclusions from this product about the importance of defense versus offense when we rate and regard superstars in the NBA.

Motivation (written from Ben's perspective who proposed the project):

The global market for the sports industry is more than 500 billion dollars in 2022, and the massive scope of this industry means it impacts nearly all of our lives in a meaningful way. Data analysis in sports is a massive and rapidly developing field, and professional sports teams have entire departments devoted to making predictions using data.

As an avid sports fan who hopes to work in the industry as a data analyst someday, a focus on professional sports was the only option for me when deciding what my project should be about. Going into this project, I knew I wanted to work with data from the NBA or AUDL (American Ultimate Disc League) in some capacity due to my own personal interests. I quickly settled on working with NBA data because it is more readily available, and because I knew that the NBA 2k video games could be invaluable as a means of quantifying individual players' skill. NBA 2k21 is the NBA's official video game, and is essentially just a game that lets you play as your favorite basketball team against your friends. Each player in the game is assigned an "overall rating," which is an all-encompassing rating of how good a player is. These ratings range from 0 to 100, with the average falling at a bit above 70. Any rating over 90 is exceptional. The usefulness of this overall rating to our project will be discussed in the problem statement section.

Problem Statement:

Our venture is first and foremost a sports prediction project where the overarching goal is to predict NBA player's overall ratings in the 2k21 video game. These predictions are based on real statistics from the 2020-2021 NBA season (i.e. points and blocks). We plan on using a linear regression model as the predictive model because this model is used in many sports prediction projects (such as those cited in the related work section). We plan on fitting three linear models,

where one is based solely on defensive statistics, one is based on offensive statistics, and the other is based on a mixed group of statistics. By comparing the predicting power of these offensive and defensive models, we hope to draw conclusions about whether defensive or offensive statistics are more important when we rate superstars in the NBA. Our final objective is to predict the NBA 2k21 overall ratings for rookies using our linear model, which could provide insight into which rookies might develop into NBA stars.

Related work (each subsection is included in the references below):

1. *Using NCAA Stats to Predict NBA Draft Order*: This group built a linear regression model to predict where future NBA players would be drafted based on college stats. This group ran a "stepwise automated selection process to let mathematics decide for us on which variables to keep and discard." [1] This notion of running hypothesis tests to see if variables really are significant ($p<.05$) is something we referenced when deciding which statistics were worthy of inclusion as predictor variables in our linear models. This project differs greatly from ours in its ultimate goal and the preprocessing steps. These two major differences also hold true between our project and related works 2 and 3 below.
2. *What makes a player score? NBA ppg predictions from a college data scientist*: This paper describes fitting a linear model to a data set of NBA players. This linear model is fit for a different purpose, to try to predict PPG and "win shares" of a player. This paper describes how altering the data itself can help when fitting a model. One example of this is how "PPG is a heavily right-skewed variable. If we log-transform PPG, we get an almost perfect fit with no assumptions violated." [2] The assumptions described in this quote are that they're working with a normal linear model (which is a useful notion also addressed in the paper).
3. *Learn linear regression using scikit-learn and NBA data: Data science with sports*: This article describes making a linear regression model to predict "box score plus minus" (a stat that describes how many points a player scores above or below the league average for 100 possessions) based on that player's individual statistics. The main application of this article for us is that it gives a demonstration of using python modules including numpy, pandas, sklearn, plotly, and streamlist for predictive modeling. This article is also useful

in that it walks through step by step how a linear model was built for a similar project to ours (including a focus on preprocessing).

Data Collection:

We knew that we needed a dataset of real NBA statistics from the 2020-2021 NBA season. The well regarded website basketballrefference.com houses a dataset including exactly this, where each statistic is on a per game basis. This dataset includes data for all players who played in the 2020-2021 season, and is readily available for download as a CSV. We then looked over our data and realized it only had two defensive variables in it. We needed more variables to base our predictive linear model on, but fortunately basketballrefference.com provided a separate table of "advanced statistics" for this same group of players from the same website, which provided a new dataset with four additional defensive variables to base our predictive models on.

We also required a dataset of NBA 2k21 OVR ratings for all players in the game. This list of players should be nearly the same group as those players in our other database because it is based on all NBA players from the 2020-21 NBA season. We began by trying to scrape the HTML from NBA2kw.com using beautifulsoup, but this website turned out to be too messy to get the players names, overall ratings, and heights (which we planned on using as an additional predicting variable in our linear regression model) from. After scouring through more NBA 2k websites online, we settled on HoopsHype.com. We downloaded HoopsHype.com as an html file, and after a brief review of beautifulsoup documentation and basic methods, we began our cleaning process in earnest.

Data Cleaning:

To clean and process our data, we used a Jupyter Notebook with pandas, numpy, matplotlib, and seaborn.

We began by downloading the two different CSV files described above from basketball reference.com. These datasets contain the same 705 players and differ in the types of statistics for each individual. Since these data frames contain the same players in the same order, we simply assigned each column from the "advanced statistics" dataframe (DWS, STL%,BLK%, and DBPM) to the other dataframe. The next step in cleaning our dataframe was to remove the

trailing 9 character ID code from the name column. This was done using a simple .str[:-10] command on the column called 'Player' that housed the names.

	Name	Pos	Tm	Age	G	GS	MP	FG	FGA	FG%	...	AST	STL	BLK	STL%	BLK%	DWS	DBPM	TOV	PF	PTS
0	precious achiuwa	PF	MIA	21.0	61.0	4.0	12.1	2.0	3.7	0.544	...	0.5	0.3	0.5	1.3	4.0	1.0	-0.5	0.7	1.5	5.0
1	jaylen adams	PG	MIL	24.0	7.0	0.0	2.6	0.1	1.1	0.125	...	0.3	0.0	0.0	0.0	0.0	0.0	-4.6	0.0	0.1	0.3
2	steven adams	C	NOP	27.0	58.0	58.0	27.7	3.3	5.3	0.614	...	1.9	0.9	0.7	1.6	2.2	1.7	0.1	1.3	1.9	7.6
3	bam adebayo	C	MIA	23.0	64.0	64.0	33.5	7.1	12.5	0.570	...	5.4	1.2	1.0	1.7	3.2	3.2	2.0	2.6	2.3	18.7
4	lamarcus aldridge	C	TOT	35.0	26.0	23.0	25.9	5.4	11.4	0.473	...	1.9	0.4	1.1	0.8	3.7	0.6	-0.2	1.0	1.8	13.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
700	delon wright	PG	SAC	28.0	27.0	8.0	25.8	3.9	8.3	0.462	...	3.6	1.6	0.4	3.0	1.3	0.5	0.6	1.3	1.1	10.0
701	thaddeus young	PF	CHI	32.0	68.0	23.0	24.3	5.4	9.7	0.559	...	4.3	1.1	0.6	2.2	2.1	2.2	1.4	2.0	2.2	12.1
702	trae young	PG	ATL	22.0	63.0	63.0	33.7	7.7	17.7	0.438	...	9.4	0.8	0.2	1.2	0.5	1.3	-1.7	4.1	1.8	25.3
703	cody zeller	C	CHO	28.0	48.0	21.0	20.9	3.8	6.8	0.559	...	1.8	0.6	0.4	1.3	1.7	1.1	-0.2	1.1	2.5	9.4
704	ivica zubac	C	LAC	23.0	72.0	33.0	22.3	3.6	5.5	0.652	...	1.3	0.3	0.9	0.7	3.4	2.1	0.4	1.1	2.6	9.0

705 rows × 33 columns

The real brunt of our cleaning and preprocessing was attempting to scrape the players NBA 2k overall ratings from the web. As mentioned above, we attempted to scrape NBA2kw.com for a long time (roughly 2 weeks) before finally moving on. This website was too difficult to scrape with beautifulsoup because many of the html tags housing information about the athletes had slight variations. Additionally, a lot of the html tags had random leading or trailing characters that made the website nearly impossible to scrape. All the beautifulsoup methods that we used to successfully scrape hoopshype.com came from our attempt to scrape NBA2kw.com

After downloading Hoopshype.com as an html file and creating a beautifulsoup object named soup, we wrote soup.prettify() to a file so we could look it over. We used beautifulsoup's find_all('tr') method to isolate the individual player entries in their respective table rows. Each entry had a player's rank, name, and overall rating (called OVR) within html tags. We started by isolating each player's OVR using the distinct class that the 'td' tags holding them had. First we used the find_all() method with a lambda function designed to find each 'td' with a specific class. We then used the rendercontents() method on each player's OVR and noticed that each entry in the array of OVR scores had leading and trailing characters. To fix this, we used the search method on each entry in the uncleaned OVR table for the regex r"[0-9]*[0-9]", and assigned the results of these regular expression searches to each row. This resulted in a cleaned array of each player's overall score as desired.

The next step was to scrape and clean the players names from the html soup. This was achieved by following the same process as described above with a couple of modifications. We used the find_all() method with a different lambda function designed to find each ‘td’ with a specific class. We then followed the process described above but used the regex r"[a-z]+-[a-z]+." The two final adjustments that we made were that no leading and trailing characters needed to be removed, and that we used the method replace("-", " ") to ensure that hyphenated names would be spelled the same way as in our table from basketballreference.com. The array of names and the array of overalls were then merged into a dataframe.

	Name	OVR
0	lebron james	97
1	giannis antetokounmpo	97
2	kawhi leonard	96
3	james harden	96
4	stephen curry	95
...	...	...
580	marial shayok	67
581	justin robinson	67
582	robert franks	67
583	antonius cleveland	67
584	kobi simmons	66

585 rows × 2 columns

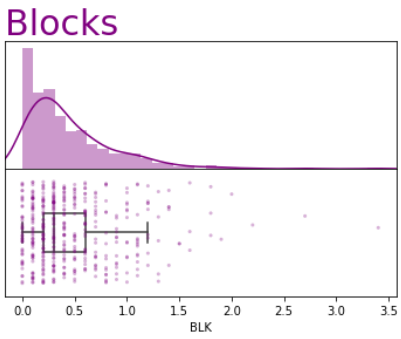
The next step in our data cleaning process was to merge the data frame of each player's real NBA statistics with the data frame of each player's overall rating. This was accomplished by an inner merge on the name variable. We lost about 130 players while performing this inner merge, likely due to spelling differences in players names between Hoopshype.com and basketballreference.com. Additionally, not every player from the basketballreference.com dataset was included in the list of OVR scores from HoopsHype.com.

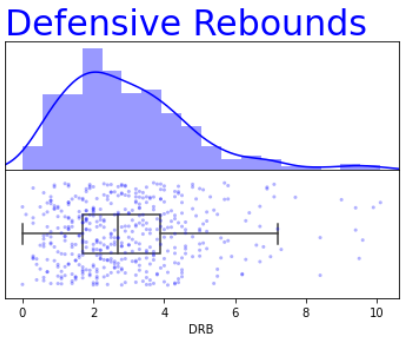
	Name	OVR	Pos	Tm	Age	G	GS	MP	FG	FGA	...	AST	STL	BLK	STL%	BLK%	DWS	DBPM	TOV	PF	PTS
0	lebron james	97	PG	LAL	36.0	45.0	45.0	33.4	9.4	18.3	...	7.8	1.1	0.6	1.6	1.5	2.6	2.3	3.7	1.6	25.0
1	giannis antetokounmpo	97	PF	MIL	26.0	61.0	61.0	33.0	10.3	18.0	...	5.9	1.2	1.2	1.7	3.2	3.3	2.8	3.4	2.8	28.1
2	kawhi leonard	96	SF	LAC	29.0	52.0	52.0	34.1	8.9	17.5	...	5.2	1.6	0.4	2.3	1.1	2.4	1.3	2.0	1.6	24.8
3	james harden	96	PG-SG	TOT	31.0	44.0	43.0	36.6	7.8	16.7	...	10.8	1.2	0.8	1.6	1.8	1.7	1.0	4.0	2.3	24.6
4	james harden	96	SG	HOU	31.0	8.0	8.0	36.3	7.5	16.9	...	10.4	0.9	0.8	1.1	1.8	0.2	-0.5	4.3	1.8	24.8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
451	jared harper	67	PG	NYK	23.0	8.0	0.0	2.0	0.0	0.5	...	0.1	0.0	0.0	0.0	0.0	0.0	-5.5	0.4	0.1	0.4
452	jarrell brantley	67	PF	UTA	24.0	28.0	0.0	4.9	0.9	1.9	...	0.5	0.3	0.1	2.5	1.2	0.2	1.6	0.3	0.6	2.3
453	gabe vincent	67	PG	MIA	24.0	50.0	7.0	13.1	1.8	4.7	...	1.3	0.4	0.0	1.6	0.3	0.5	-0.9	0.7	1.6	4.8
454	justin robinson	67	PG	OKC	23.0	9.0	0.0	9.8	0.8	2.3	...	1.0	0.3	0.0	1.6	0.0	0.0	-1.0	0.2	1.1	2.3
455	robert franks	67	PF	ORL	24.0	7.0	0.0	14.4	1.9	4.0	...	0.7	0.4	0.4	1.4	2.7	0.1	-0.8	0.3	1.1	6.1

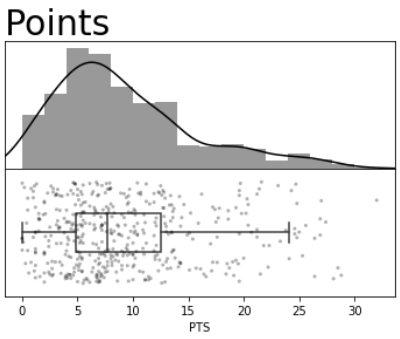
456 rows × 34 columns

Data Exploration:

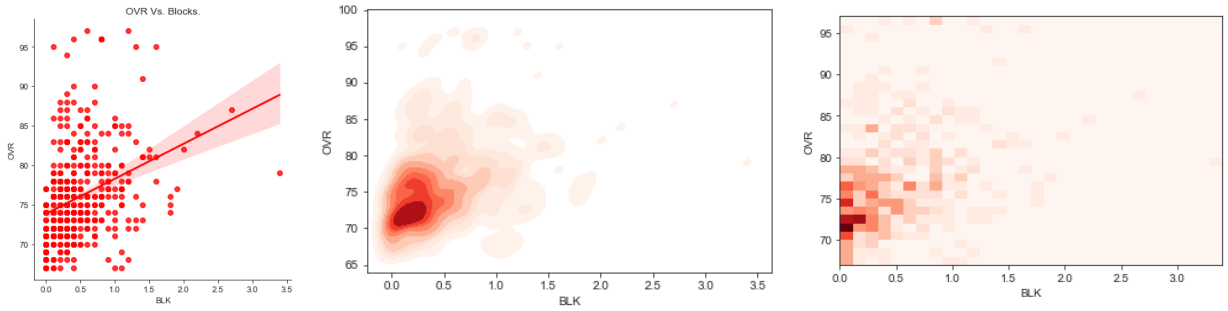
After our two datasets were cleaned and merged, we explored the data visually using seaborn (sns.distplot, sns.stripplot, sns.boxplot). We started by looking at the distributions of the individual per-game stats, like points, rebounds, blocks, and steals for all 500+ players. We noticed that all of these features were strongly skewed to the high end. This reflects how most players have a relatively low rate of scoring or blocking, but some elite players have much higher rates because they get more playing time and are the best players on their teams.



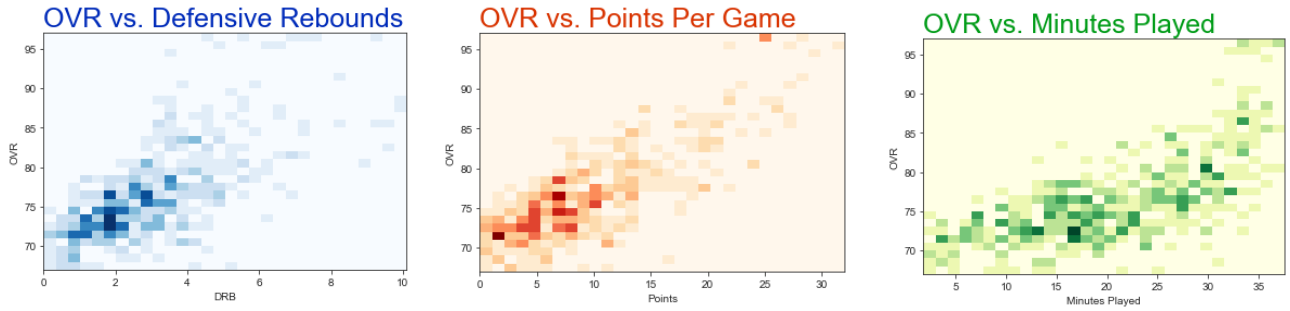




After looking at individual distributions, we looked at how each of our 30+ features were correlated with the OVR scores.

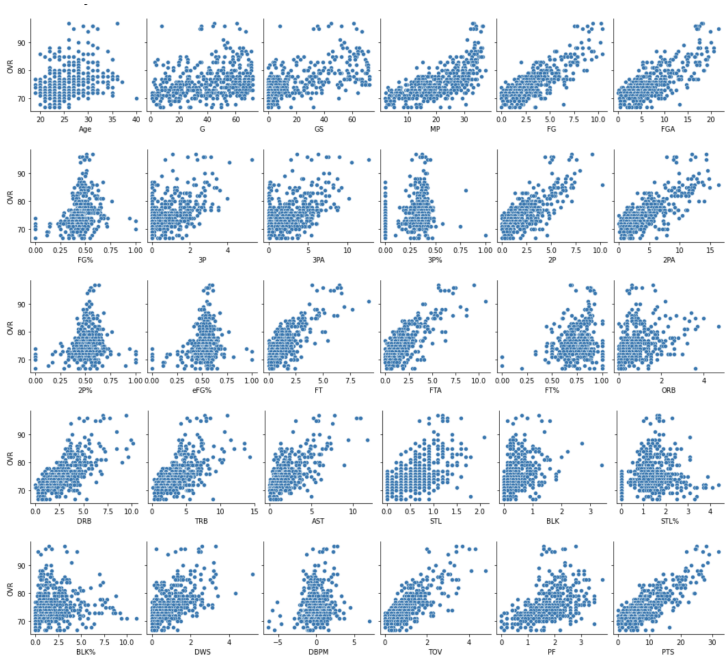


We looked at some of these relationships with more than one type of scatter plot. For example, the three plots above all show OVR score vs. Blocks per game. The first plot (a Seaborn scatterplot) clearly shows each player in the data set represented as one point, but it fails to show where scores are more heavily concentrated. The other two plots (a Seaborn density plot and a Matplotlib heat map) deal with overplotting by showing hotspots where more than one player occupies a certain stat range.



These hot spots are reflected in other features' relationships with OVR, like defensive rebounds, points per game, and minutes played.

While these heat maps are good at visualizing the most common stat ranges, we wanted to look at the linearity of each relationship. We did this by making a seaborn pair plot of OVR scores and all the other features. Some relationships were more linear than others. The features with the most linear relationships with OVR were: Minutes played, Field goals, Field goal attempts, 2 pointers, 2 point attempts, Free throws, Free throw attempts, Defensive rebounds, Total rebounds, Assists, Steals, Defensive win share, Turnovers, and Points.

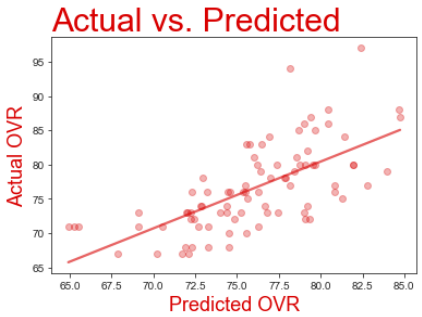


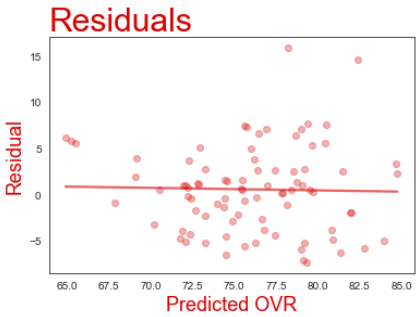
These linear relationships are what led us to choose a linear regression model for OVR prediction.

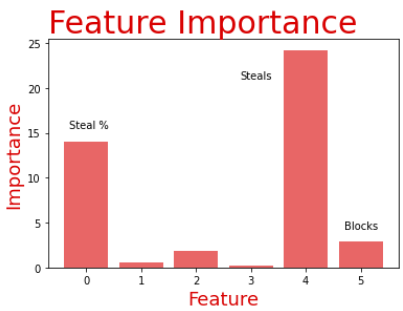
Modeling:

After we explored the data and found some evidence to support our intention of building a linear model, it was time to actually go about fitting one. We began by using scikit-learn's train_test_split method with a train_size of .8 and a random_state of 42.

We decided to fit a linear model based on 6 defensive stats first. Accordingly, we made a data frame X_train containing the dependent variables STL%, BLK%, DWS, DBPM, STL, and BLK that the model would use to predict OVR as well a data frame Y_train containing the overalls of the players in the training set (analogous X_val and Y_val data frames were made for the validation set). We then imported scikit-learn's linear_model and standard_scaler packages. We normalized X_train using standard_scalers transform functionality. In hindsight this was a misstep, as the standard scalar assumes that the data is normally distributed and we showed in the exploration of our data that our independent variables tend not to be. We then fit a linear model on the training data and predicted players OVR scores on the X_train and X_val sets.

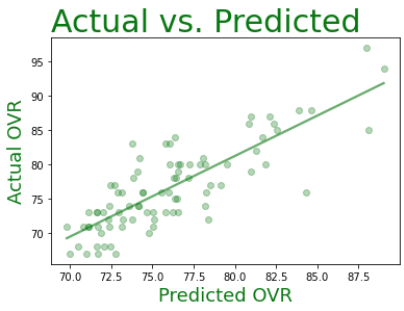


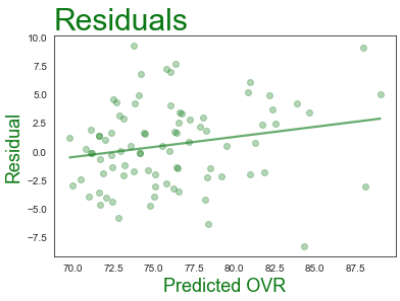


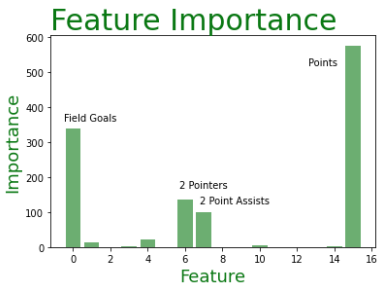


R^2 value = 0.47, **Validation error** = 4.63, **Most important features**: Steal %, Steals, Blocks.

An analogous model was fitted that predicted players' overall ratings based on 16 offensive statistics. The training set for this model included variables: 'FG', 'FGA', 'FG%', '3P', '3PA', '3P%', '2P', '2PA', '2P%', 'eFG%','FT', 'FTA', 'FT%', 'ORB', 'AST', and 'PTS.'

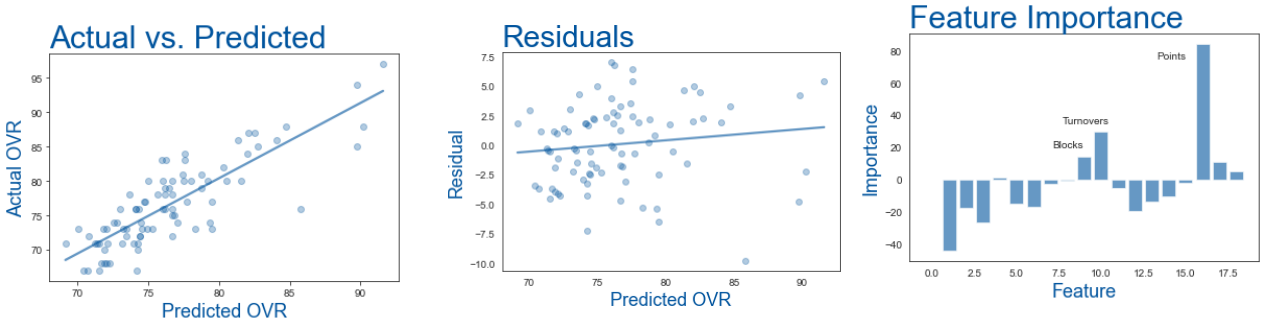






R^2 value = 0.70, **Validation error** = 3.64, **Most important features:** Points, Field goals, 2 pointers, 2 point attempts.

Yet another analogous model was fitted that predicted players' overall ratings based on an assortment of 19 offensive and defensive statistics. The training set for this model included variables: 'G', 'FG', 'FGA', '3P', '3PA', 'FT', 'FTA', 'ORB', 'STL', 'BLK', 'TOV', 'PF', 'FG%', '3P%', 'FT%', 'MP', 'PTS', 'TRB,' and 'AST.'



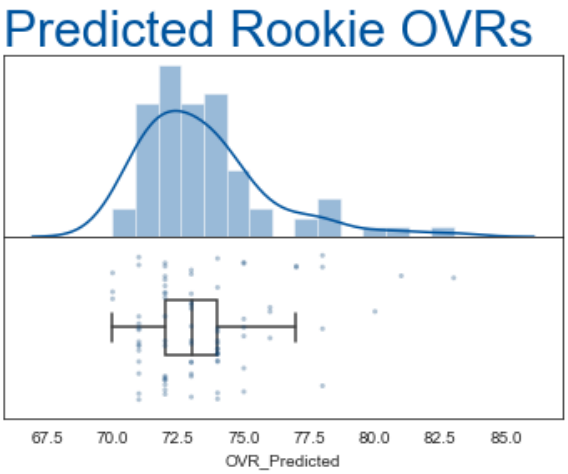
R^2 value = 0.74, **Validation error** = 3.40, **Most important features:** Points, Turnovers, Blocks.

After we finished training our three models, we wanted to use the best one to actually predict some unknown OVR scores. We used the model that was trained on the 19 mixed features because it had the highest R^2 value and the lowest validation error.

Rookies Career BAA/NBA Stats Share & Export ▾ Glossary

					Totals																Shooting			Per Game			
Rk	Player	Debut	Age	Yrs	G	MP	FG	FGA	3P	3PA	FT	FTA	ORB	TRB	AST	STL	BLK	TOV	PF	PTS	FG%	3P%	FT%	MP	PTS	TRB	AST
1	Precious Achiuwa	Dec 23, '20, MIA @ ORL	21	1	61	737	124	228	0	1	56	110	73	208	29	20	28	43	91	304	.544	.000	.509	12.1	5.0	3.4	0.5
2	Ty-Shon Alexander	Dec 27, '20, PHO @ SAC	22	1	15	47	3	12	2	9	1	2	2	10	6	0	1	3	2	9	.250	.222	.500	3.1	0.6	0.7	0.4
3	Cole Anthony	Dec 23, '20, ORL vs. MIA	20	1	47	1273	219	552	58	172	109	131	38	221	192	30	18	106	98	605	.397	.337	.832	27.1	12.9	4.7	4.1
4	Deni Avdija	Dec 23, '20, WAS @ PHI	20	1	54	1257	130	312	53	168	29	45	22	262	63	32	15	33	140	342	.417	.315	.644	23.3	6.3	4.9	1.2
5	Udoka Azubuike	Dec 23, '20, UTA @ POR	21	1	15	57	4	9	0	0	8	10	4	13	0	1	4	3	9	16	.444		.800	3.8	1.1	0.9	0.0
6	LaMelo Ball	Dec 23, '20, CHO @ CLE	19	1	51	1469	293	672	92	261	125	165	63	302	313	81	18	145	136	803	.436	.352	.758	28.8	15.7	5.9	6.1
7	Desmond Bane	Dec 23, '20, MEM vs. SAS	22	1	68	1519	234	499	117	271	40	49	31	210	118	41	16	59	125	625	.469	.432	.816	22.3	9.2	3.1	1.7
8	Saddiq Bey	Dec 26, '20, DET vs. CLE	21	1	70	1909	279	691	175	460	124	147	43	318	95	52	14	60	110	857	.404	.380	.844	27.3	12.2	4.5	1.4
9	Tyler Bey	Jan 13, '21, DAL @ CHO	22	1	18	71	7	22	1	4	3	5	8	19	3	0	1	3	6	18	.318	.250	.600	3.9	1.0	1.1	0.2
10	Keljin Blevins	Dec 23, '20, POR vs. UTA	25	1	17	75	5	20	2	8	0	0	3	10	4	2	0	5	8	12	.250	.250		4.4	0.7	0.6	0.2
11	Amida Brimah	Apr 25, '21, IND @ ORL	26	1	5	29	5	8	0	0	3	3	2	8	1	0	5	4	4	13	.625		1.000	5.8	2.6	1.6	0.2
12	Armoni Brooks	Apr 9, '21, HOU @ LAC	22	1	20	520	78	192	60	157	7	12	10	68	30	12	5	22	34	223	.406	.382	.583	26.0	11.2	3.4	1.5
13	Elijah Bryant	May 16, '21, MIL @ CHI	25	1	1	32	6	13	1	5	3	3	2	6	3	0	1	4	4	16	.462	.200	1.000	32.0	16.0	6.0	3.0
14	Facundo Campazzo	Dec 23, '20, DEN vs. SAC	29	1	65	1425	120	315	76	216	80	91	22	134	232	79	14	73	132	396	.381	.352	.879	21.9	6.1	2.1	3.6
15	Devin Cannady	Apr 7, '21, ORL vs. WAS	24	1	8	74	11	28	6	16	6	7	0	5	1	5	1	2	9	34	.393	.375	.857	9.3	4.3	0.6	0.1
16	Vernon Carey Jr.	Dec 30, '20, CHO @ DAL	19	1	19	115	18	36	1	7	9	11	6	27	2	1	5	5	13	46	.500	.143	.818	6.1	2.4	1.4	0.1

We downloaded a CSV of 94 rookies from the 2020-21 NBA season and used our linear model to predict their OVR scores.



Min: **70**

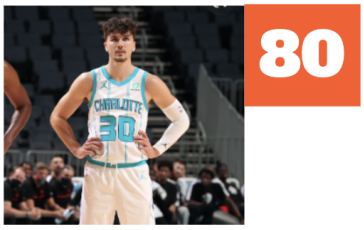
Max: **83**

Mean: **73.4**

The distribution of the rookies' OVR scores was similar to the other 500+ players because it was skewed to the high end, but it had a lower mean and a smaller range. The highest predicted OVR score from this dataset was 83. This makes sense because this was being used on a dataset of rookies, who don't have stats as good as the most elite NBA players yet.

Keljin Blevins

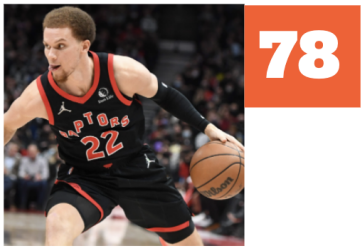
Udoka Azubuike



Nate Darling

Jaden Mcdaniels

Jae'Sean Tate

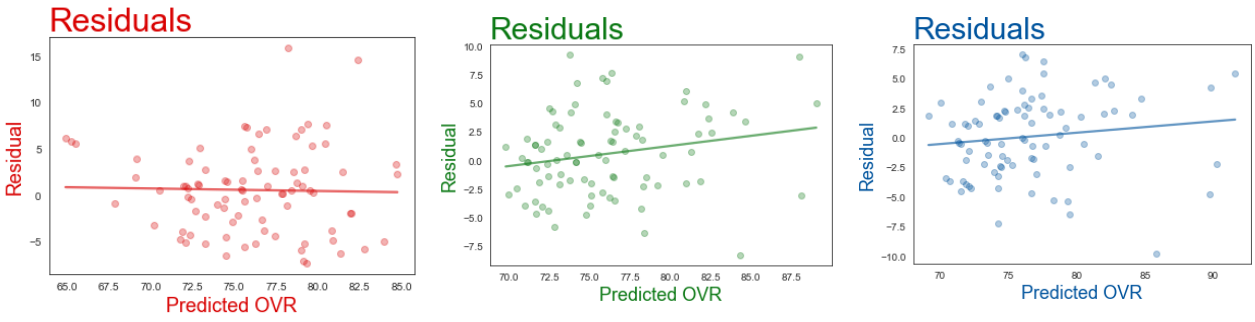


Malachi Flynn

There were six high outliers. Above are the 2020-21 rookies with the highest predicted OVR scores.

Analysis and results:

Since our residual plots do not show any discernible linear relationship, we can conclude that a linear model was a good choice to fit this data.



According to our model, we can confidently say that defensive statistics like steals and blocks have very little influence on the calculation of players' OVR scores in NBA 2k21.

Model Fit

	Defense	**Offense**	**Mixed**
R^2	0.47	0.70	0.74
Training Error	4.01	3.02	2.81
Validation Error	4.63	3.64	3.40

The model that used just defensive features had the lowest R^2 value at 0.47 (fairly weak), as well as the highest training and validation error. The model that used just offensive features had a much higher R^2 value at 0.70 (moderate) and lower errors. Our best model used 19 offensive and defensive features, and had the highest R^2 value at 0.74 as well as the lowest errors.

Permutation Importance

Defense	Perm. imp.	**Offense**	Perm. imp.	**Mixed**	Perm. imp.
Steals	24.23527	Points	576.25272	Points	83.99036
Steal %	13.98909	Field Goals	340.02493	Turnovers	29.76969
Blocks	2.92047	2 Pointers	137.05598	Blocks	14.30556
DWS	1.82538	2 PA	98.61912	Total Rebounds	10.80018

To identify which features were the most important within each model, we ran permutation tests. This test randomly shuffles each label in the dataset to break the ties between the predictor and its predicted value. It then compares the percentage of random trials that had relationships similar to the original unshuffled data. If the proportion is similar, the feature has low importance.

For our defensive model, Steals and Steal% were the most important predictors of OVR score. For our Offensive model, Points and Field goals were the most important predictors. These offensive stats had permutation test scores over 20 times higher than the features in the defensive model. In our mixed model, Points again had the highest permutation importance, followed by Turnovers, Blocks, and Total Rebounds.

We then performed hypothesis testing on both our offensive and defensive linear models as a further means of demonstrating different features predicting powers. The hypothesis test suggests that the STL%, DWS, STL, and BLK variables are significant predictors (with a p-value less than .05) of a players OVR score. DWS is a stat designed to show how much a player contributes overall on defense, and how much that individual helps their team win; so it's sensible that it would have such significant predicting power for OVR score. Our hypothesis test found that the most significant offensive statistics when predicting OVR were 3P%, AST, PTS, FG, 2P, and FT%. That said, of this group the only variables with p-value under .1 were AST and 3P%; and the only p value under .05 was AST.

Future work:

While reflecting on our project we came up with three ways that we could improve on our project and continue to work on it going forward.

The first step we could take would be to use data from a NBA season other than 2020-21. This would let us evaluate the model on a dataset that should resemble ours, but our model wasn't fitted on.

A third way to continue working on our project would be to use rookies statistics from a few years ago and try to predict these players' overalls using our model. These players would have already had time to develop in the NBA; and we could compare the players our model predicted would have a high overall with those who actually panned out.

One final addition to our project would be to attempt to predict players positions based on their NBA statistics. This would be a different path to pursue then what our projects focussed on to this point, but we believe it would be a worthwhile pursuit as classification problems are something both of us are interested in learning more about.

OVR score prediction can also be applied to other professional sports video games, such as FIFA for soccer or NHL for hockey.

Conclusion:

Our analysis has led us to conclude that a linear model was an appropriate method to predict players' overall scores in NBA 2k21 based on their statistics in the 2020-21 NBA season.

We have some evidence to suggest that offensive variables such as points or two point attempts hold more predictive power for a player's NBA 2K21 rating than defensive variables such as steals and blocks.

Our results are in line with what you see from the sports prediction field as a whole. For example, points scored is a powerful predictor of a players abilities in basketball (and sports as a whole). However, our model also shows that sports predictions are often unreliable. For instance, looking at famous draft busts such as Anthony Bennett, who performed extremely well in college but did not do well in the NBA. Finally, our models predictions of rookies' overall ratings should be taken with a grain of salt due to the limited predicting power we've shown this linear model to have.

References:

1. Brown, George, et al. “Using NCAA Stats to Predict NBA Draft Order.” *Playing Numbers*, 4 Mar. 2020, https://www.playingnumbers.com/2020/03/using-ncaa-stats-to-predict-nba-draft-order/#:~:text=This%20model%20.
2. Liu, Michael. “What Makes a Player Score? NBA PPG Predictions from a College Data Scientist.” *Medium*, Medium, 30 June 2020, https://medium.com/@michaelliu36/what-makes-a-player-score-nba-ppg-predictions-from-a-college-data-scientist-3086f9fbb23.
3. Hwang, JP. “Learn Linear Regression Using Scikit-Learn and NBA Data: Data Science with Sports.” *Medium*, Towards Data Science, 18 Sept. 2020, https://towardsdatascience.com/learn-linear-regression-using-scikit-learn-and-nba-data-data-science-with-sports-9908b0f6a031.
4. “2020-21 NBA - Season Stats.” *Basketball*, https://www.basketball-reference.com/leagues/NBA_2021_rookies-season-stats.html.
5. Update, NBA 2K. “NBA 2K Update.” *NBA 2KW NBA 2K22 Locker Codes NBA 2K22 News NBA 2K22 MyPLAYER Builder NBA 2K22 Tips NBA 2K22 Ratings NBA 2K Community NBA 2K23 News NBA 2K23 Wishlist*, 8 Sept. 2021, https://nba2kw.com/list/nba-2k21-all-player-ratings/.
6. “2020-21 NBA Player Stats: Per Game.” *Basketball*, https://www.basketball-reference.com/leagues/NBA_2021_per_game.html.