Data Science Union: The NBA Project

PROJECT OVERVIEW

Each year, a player's annual salary is determined by the player's impact, which is measured by various factors: points per game, number of rebounds, 3-point field goal percentage, and so forth. Our goal was to create a single dataset with players' salaries along with their statistics over the years, determine the most influential factors on a player's salary, and then leverage machine learning to determine an algorithm that will enable us to best predict a player's annual salary based on these statistics. Our main language for this project was Python.

DATA PREPROCESSING

Our data sources for this project were Basketball Reference, which we used to obtain individual players' game statistics per season (i.e. average number of points per game), and Hoops Hype, which we used to obtain the individual players' salaries per season. As the NBA determines a player's salary for that season prior to the start of the season, we made the assumption that the player's performance in the season prior would be used to determine the player's salary the following season. Using a combination of Excel and Python, we mapped the player's game statistics to their respective salaries, ultimately merging the players' seasonal statistics data set with the players' salaries dataset from the following season. We also added the columns:

StartYr - the season's start year from which the statistics were recorded
EndYr - the season's end year from which the statistics were recorded
SalStartYr - the following season's start year (also the season when player is paid)
SalEndYr - the following season's end year (also the season when player is paid)

Using python, we merged all the seasonal data sets with the salaries and statistics from the year 1980 on. However, we could not use 1980-1989 data as there was no record of the players' salaries on Hoops Hype.

Working on Google Colab and using Python (specifically the Pandas package), we worked on preprocessing the now merged data set with all the players' statistics and salaries. We subsetted the data set to only include the years 2000 to 2020 on as we wanted the data to be as current as possible. We later decided to use the years 1990 to 2020. Furthermore, we dropped the rows with missing values.

After doing some exploratory data analysis and modeling, we added the column *years of experience* by counting the number of times a player appeared in the data set. For instance, a player who only appeared once would have 1 year of experience.

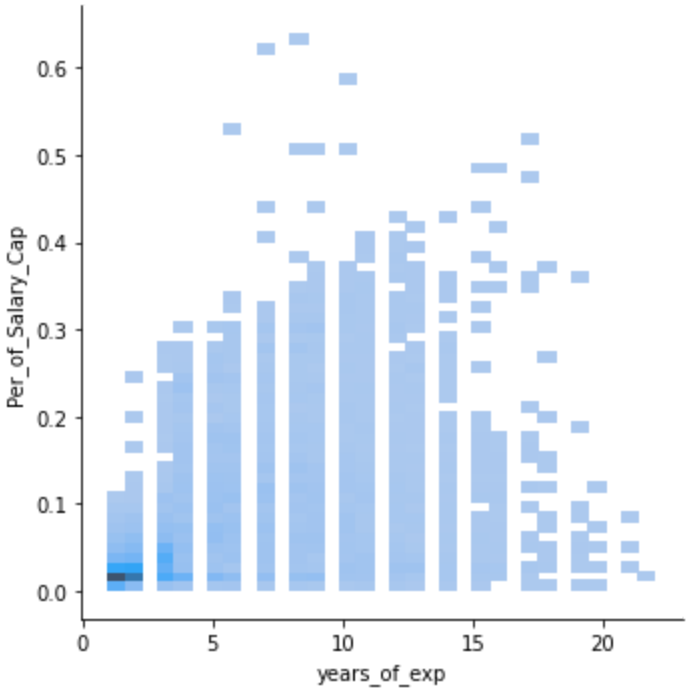
Towards the end of our project, we also decided to add the column *Salary from the previous season* to the players who appeared more than once in the dataset.

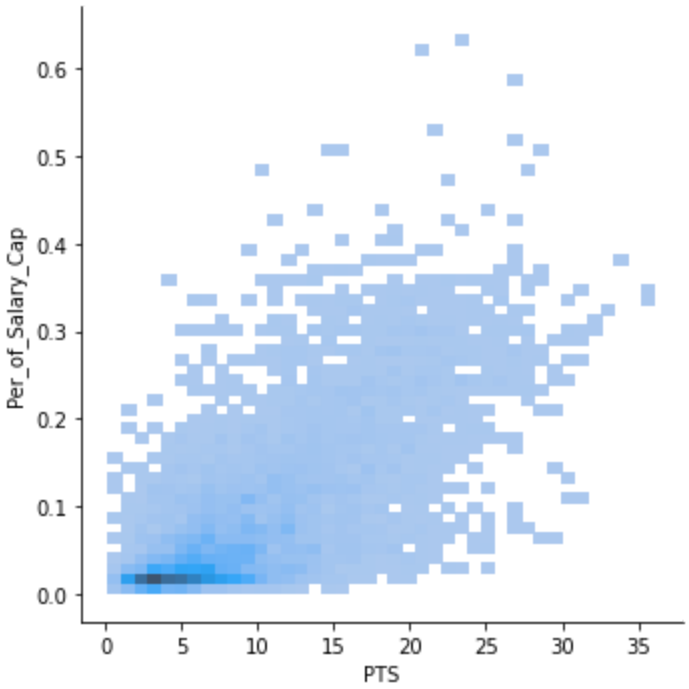
Finally, we tried two different target variables. In our initial regression and classification models, we decided to normalize the *Salary* variable by dividing by *Salary Cap*. Our reasoning behind this was because *Salary Cap* changes each season (a team's salary cap is the amount of money a team has to spend on its players), we would predict the percentage of the salary cap a player would be paid. In later approaches, we predicted purely salary as it seemed that the year accounted for the change in salary cap.

EXPLORATORY DATA ANALYSIS

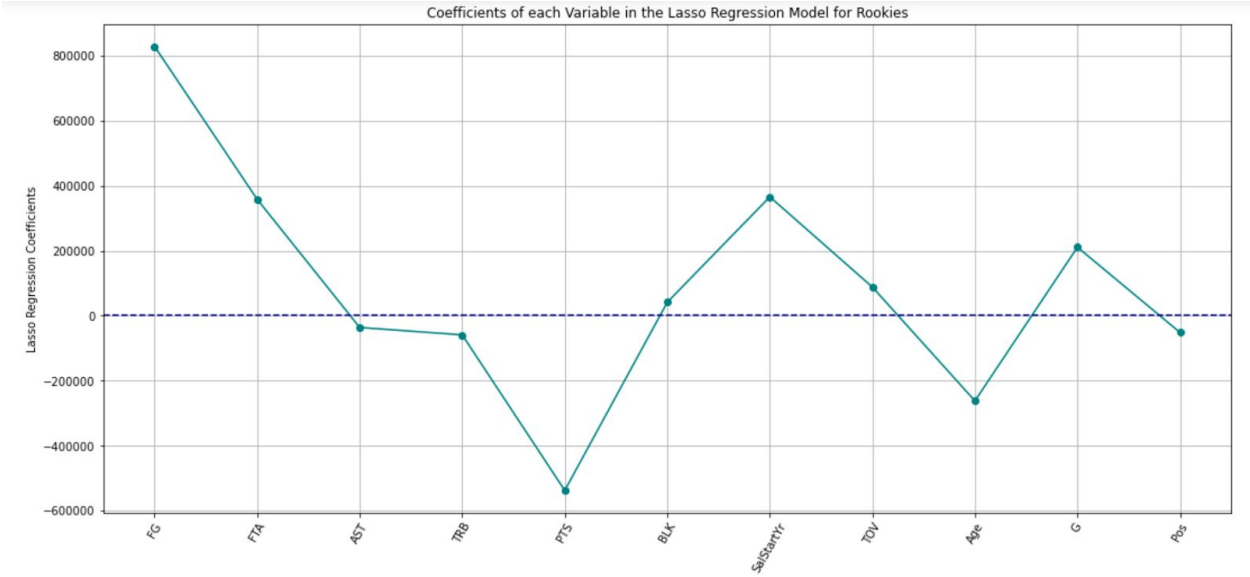
Over the course of this project, we tried several different models, both regression and classification. As different people were delegated to do different models, we had various visuals that both contributed to the intended model and contributed to the overall project as well.

The main goal of our exploratory data analysis was to determine the best predictors (and best combination of predictors) that we could use to predict a player's normalized salary or salary. We first used a heat map to provide us with a general idea of the predictors that were most highly correlated with salary.



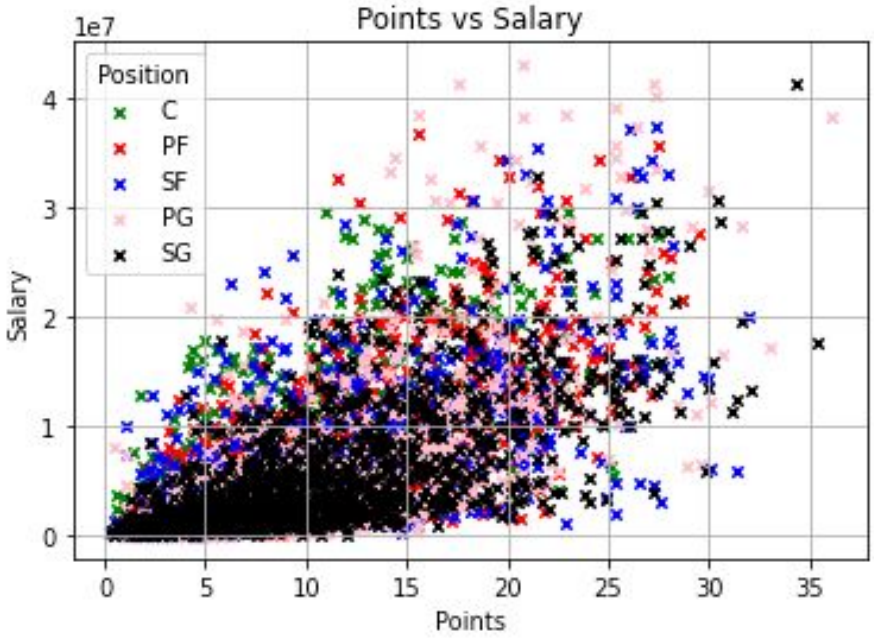


Visual: Used in Lasso Regression

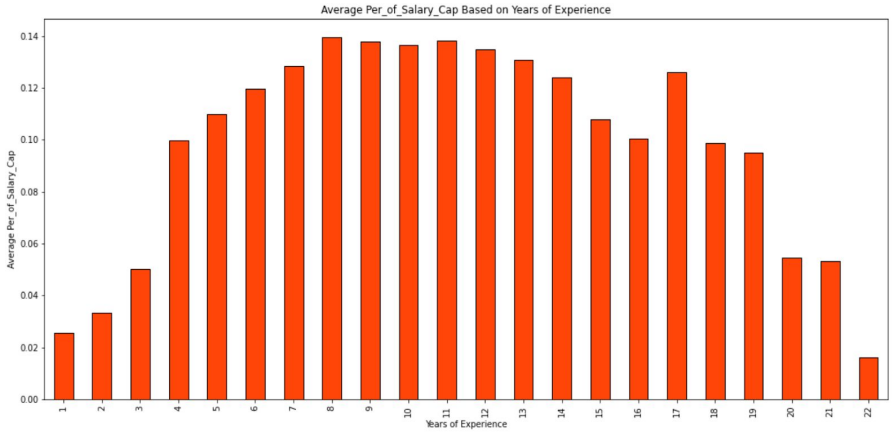


The use of lasso regression (and this visual) also helped point us towards a subset of variables that we could use for our model. Another method that we used to find a subset of variables for our model was forward selection in R.

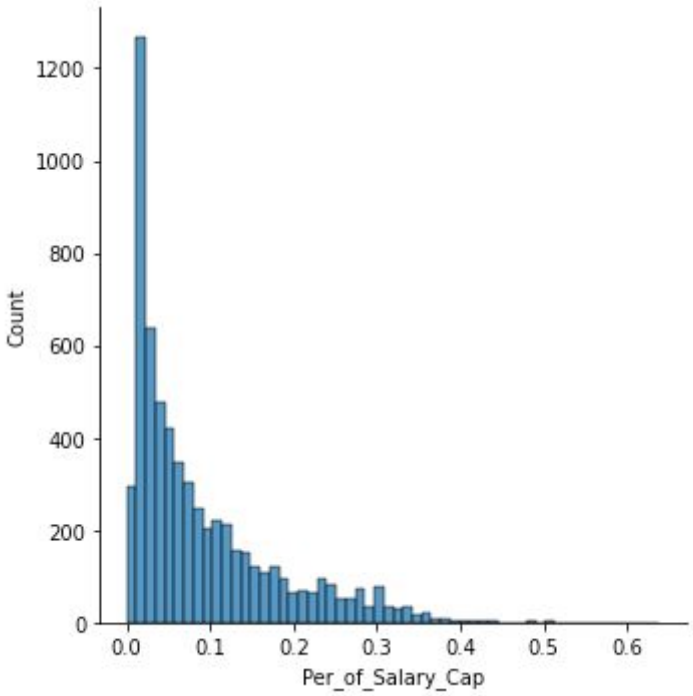
After doing some broad analysis of variables, we delved deeper to see if there was correlation between our predictor variables along with the strength of the correlation between individual variables and our target variable.



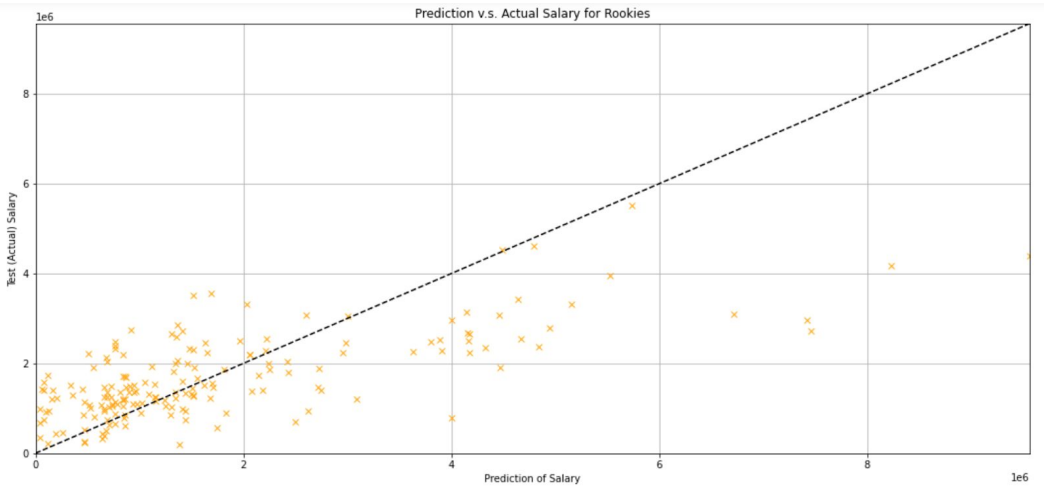
This image shows the relationship between points and salary, color coded by the position of the player. In this graph, we see that there is a generally positive correlation between points and salary, which corroborates our findings from our previous visuals. Furthermore, we see that the *position* variable may not be a strong predictor for our target variable Salary.



One column or variable that we added later on in the process was *years of experience.* After adding this variable, we observed a fairly strong positive correlation between *years of experience* and both salary and normalized salary. However, upon looking at this visual, we did also observe a peak and then a dip, so there is a point where an increase in years of experience resulted in a lower salary. A reason for this dip in salary as years increase is due to the fact that a players' older age may impact their performance.



This visualization brings to light the fact that there are a large number of players that are paid in a certain range or amount; as anticipated, there are a relatively small number of players paid higher than 0.2 and an even smaller number of players paid more than 0.3 of the salary cap.



This visual was a visual for lasso regression. Like the visual prior, we notice that there are a large cluster of players towards the bottom of the fitted line. We then see the higher paid players' actual salaries dip away from the line.

From the last two visuals, we were able to see that our model would somehow have to account for the imbalance in players' salaries. As a majority of players are paid in a similar

salary range and the “all stars” or more well-known players are a relatively smaller group, we would have to somehow account for the imbalance in data and possibly outliers.

This section does not cover all of our data visualizations: it simply covers the major observations we made. We further cover in detail our visualizations for each model in the following sections.

MACHINE LEARNING MODELS

REGRESSION

Ridge Regression

Ridge Regression is similar to standard linear regression, but it adds a penalty to the weight matrix, performing L2 regularization. This helps alleviate the effects of interdependence of different variables. In our dataset, it was anticipated that different variables would be highly correlated. For example, years of experience would probably yield higher points, assists, etc. due to the added experience. For this reason, Ridge Regression seemed like a good idea to try. Overall, the results were not very good. The R-squared score was 0.272. R-squared generally ranges from 0-1, 0 meaning no learning occurred, 1 meaning the model has learned perfectly. Note: the R-squared score can be negative. It essentially means the same as 0 (nothing learned). When testing on the rookie dataset, the RMSE score was 1,190,732 which is also quite high considering that averages were in the low millions.

Lasso Regression

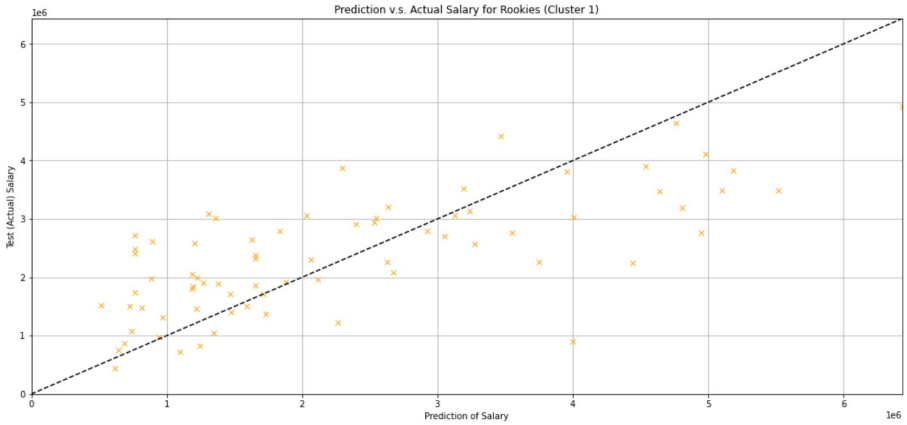
To predict Salary (and Per_of_Salary_Cap initially), we tried using lasso regression because of its regularization and feature selection properties. Lasso regression is very similar to ridge regression in that it adds a penalty value to the simple linear regression model to prevent overfitting. Lasso regression is also useful for feature selection because it is able to reduce coefficients of insignificant variables in the model to 0.

For the most part, lasso regression was not very effective at predicting Salary. With our best subset of predictors (FG, FTA, AST, TRB, PTS, BLK, SalStartYr, TDV, Age, G, Pos), lasso regression had mediocre performance at best, predicting rookie salaries with an R^2 score of 0.47615 and RMSE of 1214715.13122, and predicting veteran salaries with an R^2 score of 0.59326 and RMSE of 3067946.81366. It performed even worse for clusters within the rookie and veteran groups, with most tests never reaching an R^2 score above 0.2 and having a very high RMSE. The one exception was Cluster 1 of the rookies, which had an R^2 score of 0.50758 and an RMSE of 1033420.19457. Lasso regression was ultimately not as

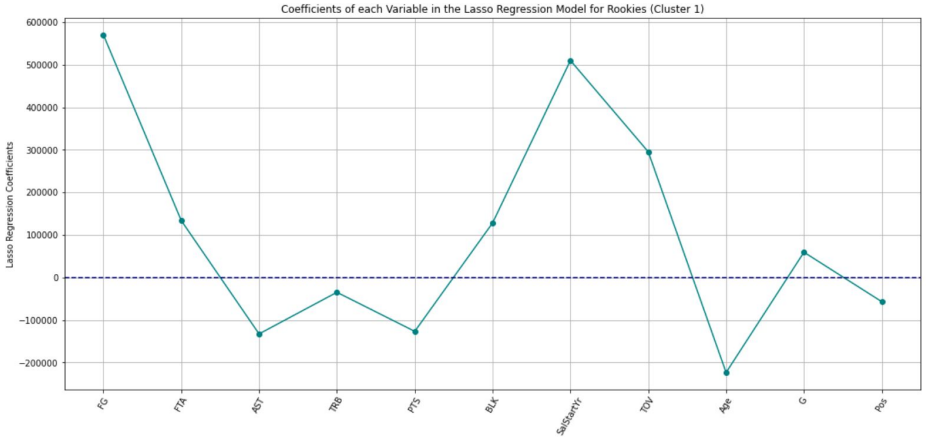
successful at predicting Salary as classification models we used, but it did help point us to which variables were best at predicting Salary.

	Rookies	Veterans
Overall	**R^2:** 0.47615 **RMSE:** 1214715.13122	**R^2:** 0.59326 **RMSE:** 3067946.81366
Cluster 0	**R^2:** -0.08045 **RMSE:** 673051.01480	**R^2:** 0.20333 **RMSE:** 1526392.85575
Cluster 1	**R^2:** 0.50758 **RMSE:** 1033420.19457	**R^2:** 0.17428 **RMSE:** 1092722.81674
Cluster 2	**R^2:** -0.02083 **RMSE:** 880597.34830	**R^2:** 0.18218 **RMSE:** 2340873.13229

Predicted v.s. Actual Salary of Rookies (Cluster 1)



Coefficients of Predictors for Salary of Rookies (Cluster 1)



Note: The blue dashed line at the x-axis means that the coefficient of the variable is 0. If a variable's coefficient is 0, this means that the variable is insignificant.

Gradient Boost Regression

Gradient boost regression is a type of machine learning boosting model. This means that it uses several different models, with each successive model trying to correct the errors of the last. Gradient boosting aims to predict the target outcomes for the next model to minimize error. The target outcomes are based on the gradient of the error with respect to the prediction.

With regards to our project, gradient boost regression was more accurate when we were predicting the percent of salary cap. The predictors that were used *years_of_exp, MP, GS, PTS, FG, AST, STL, BLK, FTA.* And, with gradient boosting, we had an R^2 score of 0.6750. But, when we switched to predicting actual salary, gradient boost regression did not prove to be useful. The results when predicting salary were worse than predicting percent of salary cap, regardless of the predictors used.

Linear

Initially, we tried linear and polynomial regression using all the predictors to see results. We had two different analysts try linear and polynomial regression.

The results from the first linear and polynomial regression models we tried are shown below. This used all players (rookies and veterans) to predict a percent of the total salary cap. Our validation results were:

Linear Regression: R2 score .597
Polynomial Regression (Deg 2): R2 score 0.627
Polynomial Regression (Deg 3): R2 score -10.9

Choosing the best (degree 2), the test results were an R2 score of 0.66 with an RMSE of 0.052. It should be noted that because this was predicting percentages, the RMSE is much lower.

The results from the second linear and polynomial regression models we tried are also shown below.

Predicting Per_of_Salary_Cap using linear regression

- R^2: 0.579
- RMSE: 0.05739

Predicting Salary with clusters using Linear Regression

	Rookies	Veterans
Cluster 0	**R^2:** 0.22675699952561645 **RMSE:** 1,037,152	**R^2:** 0.21381234061244714 **RMSE:** 1,054,213
Cluster 1	**R^2:** 0.5026136526107831 **RMSE:** 1,143,150	**R^2:** 0.18116383581789108 **RMSE:** 1,513,386
Cluster 2	**R^2:** 0.12185637634530422 **RMSE:** 6,560,329	**R^2:** 0.12185637634530422 **RMSE:** 6,560,329

Predicting Salary with clusters using polynomial regression

	Rookies	Veterans
Cluster 0	**R^2:** -0.2496593357762535 **RMSE:** 1,318,500	**R^2:** -0.0693122361034550 **RMSE:** 1,034,125
Cluster 1	**R^2:** 0.1681826815328873 **RMSE:** 1,478,326	**R^2:** -0.0207837259673038 **RMSE:** 1,689,733
Cluster 2	**R^2:** 0.09191768830086877 **RMSE:** 1,194,593	**R^2:** -0.0693122361034550 **RMSE:** 2,742,530

MACHINE LEARNING MODELS
CLASSIFICATION

We also wanted to attempt some pure classification models in order to predict salary groupings rather than a specific number. Starting with our cleaned data set, we first used clustering in order to cluster the salary data and see if there was any relationship between players' stats and their respective salary cluster. In order to find the optimal number of salary clusters, we used the elbow method, testing values 1 - 10 and found 4 to be the optimal number of clusters. Thus, each row had a cluster number (0,1,2,3) which we appended to the data set.

From the above graph, we see that the point of inflection or "elbow" occurs at k = 4. From here, we then implemented some classification models to see how well we could predict cluster values (0,1,2,3), which corresponds to a range of salary values. In other words, we

wanted to use player data/statistics to predict what salary cluster they might end up in. We implemented 4 models: logistic regression, decision tree, random forest, and support vector machine. The score for the classification models were as follows:

	Rookies	**Veterans**
Model		
Logistic Regression	**Training score:** 62.2 **Testing score:** 51.1	**Training score:** 58.4 **Testing score:** 54.3
Decision Tree	**Training score:** 58.6 **Testing score:** 51.7	**Training score:** 57.7 **Testing score:** 53.2
Random Forest	**Training score:** 65.0 **Testing score:** 52.2	**Training score:** 74.5 **Testing score:** 55.8
Support Vector Machine	**Training score:** 60.5 **Testing score:** 50.5	**Training score:** 59.1 **Testing score:** 55.6

From the above scores we notice that they are consistently around high 50s with some slight evidence of overfitting in the data. These results taught us that we needed more data in order to better predict salary as each model would max out at these scores even after using cross validation. These findings led us to our next idea which involved adding previous year salary for each veteran in the original data set. Since we earlier found that not any one player statistic directly correlated with salary, then adding previous year salary would be a good indicator as to what a player would make the following year. After adding previous year salary to the data set and redoing the above steps, we achieved much better results (around 90% for Veterans for each model for each score and near perfect for Rookies for each model and each score). These steps were implemented in our final model in which we used both classification and regression models.

MACHINE LEARNING MODELS
CLASSIFICATION WITH REGRESSION

After evaluating all our models, we came to the conclusion that classification with regression would yield the best outcome. Having grouped our data by players with years of

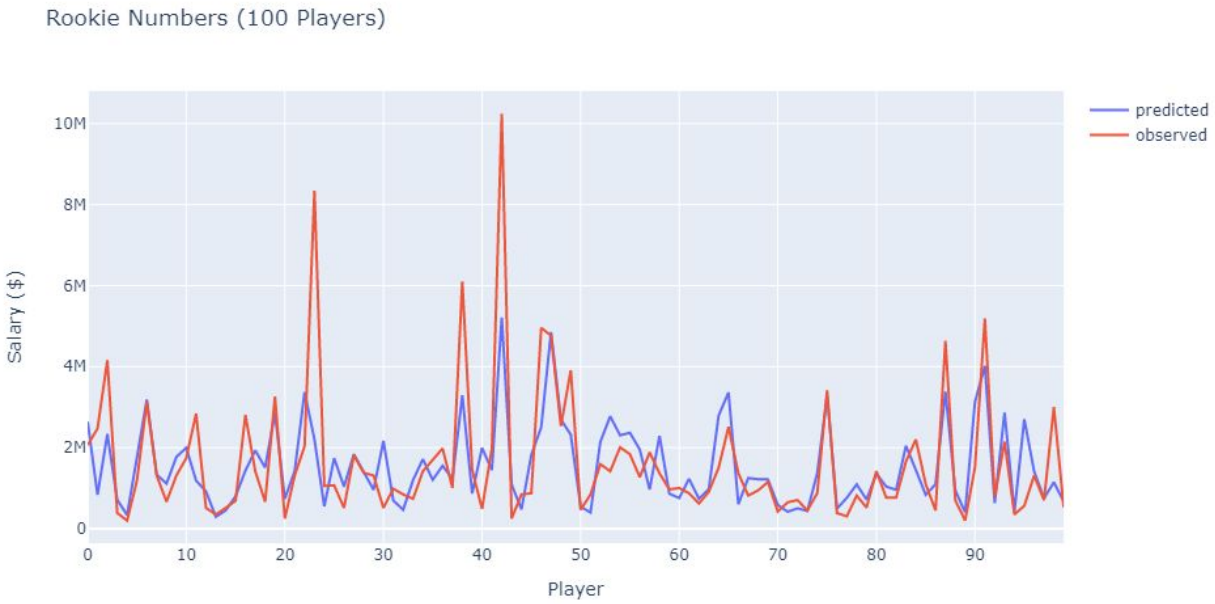
experience equal to 1 (i.e. Rookies) and experienced players (i.e. Veterans), we proceeded to further create subgroups in an effort to reduce variation. By using K-Means clustering, we achieved this task and landed on creating three separate sub groups across both the Rookie and Veteran subsets.

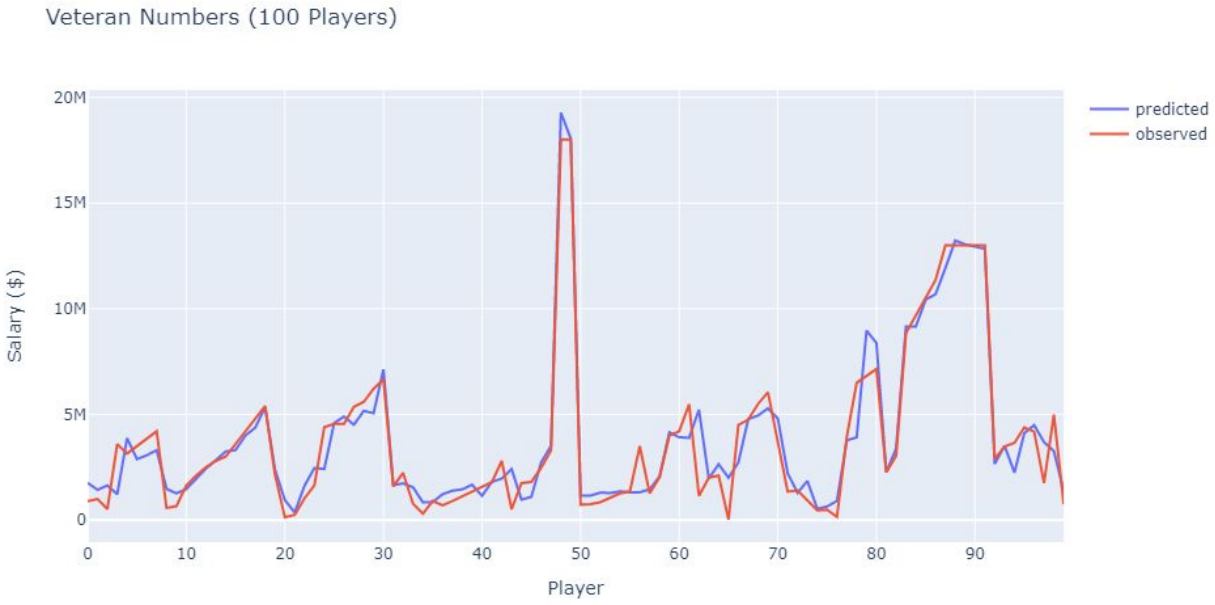
Based on previous feature selection from earlier, we knew what subsets of columns could be considered good predictors for both the Rookie and Veterans subsets. At this point in time, we decided to transition from pure regression techniques to XGBoost. XGBoost is a decision-tree based algorithm that incorporates gradient boosting. What this means is that XGBoost minimizes errors in sequential models, and in turn is well tuned to medium data that is structured or tabular - like our NBA dataset.

Once more we use RMSE as a measure of how well our model is predicting salaries. At the end of the day, RMSE is simply a standard way to measure the error in a model. Heuristically, RMSE can be considered to be the "normalized distance" between for example a vector of observed values and a vector of predicted values. After optimizing hyperparameters with XGBoost in an effort to decrease RMSE, we ended with the following results:

	Rookies	**Veterans**
Overall	**RMSE:** 968145.64	**RMSE:** 866313.68
Cluster 0	**RMSE:** 989329.91642	**RMSE:** 1021782.27539
Cluster 1	**RMSE:** 920915.46184	**RMSE:** 1617855.69241
Cluster 2	**RMSE:** 868250.72518	**RMSE:** 2519363.49779

The results of XGBoost can be illustrated in the below plots which look at how well our predicted results line up with the actual observed values. On the x-axis, we have a sample of 100 players from each of the subsets (Rookies and Veterans). On the y-axis, we have their salary: both predicted by our model and the actual observed value.





From the above plots, it's clear that our Veteran model typically outperforms our model for Rookie players. The reason for this discrepancy can likely be attributed to the fact that our Veteran model takes into account a player's salary the season before when predicting a player's salary for the next season. The Rookie model does not have this luxury seeing as Rookie player salaries are modeled based on raw stats alone and not their salary from a previous year.