Classifying NBA All Stars

Introduction

For my data mining task, I chose to investigate what differentiates National Basketball Association (NBA) All-Stars from the rest of the players. NBA All-Stars are largely considered to be the “best” 24(ish) players in the NBA for a given season. Players have historically been selected only by fan voting, but in 2017 players and media were given equal splits of the voting power so that fans only had 50%. The NBA is split into two conferences, and 12 players are selected from each conference, with starting lineups for each conference consisting of 2 backcourt players (guards) and 3 frontcourt (forwards and centers).

I first had the idea for this project after learning about the Perceptron classifier in my Introduction to Data Mining course at Washington State University. In this class, I first learned how a Perceptron could be trained to classify incoming emails as spam by looking at different characteristics of the incoming email. I was amazed at the simplicity of this algorithm, as prior to the class, I always looked at all of Machine Learning as an incredibly deep and complex task that I never thought I would understand. After Perceptron, we learned about other, complex, Machine Learning Algorithms (MLA), including Support Vector Machines (SVM), Decision Trees, and ensembles. I learned how the ensemble, XGBoost, won Kaggle Dataset Challenges, by combining many different learners into one model, and that really interested me.

When I started my project, I had a few questions in mind in which I wanted to find an answer for. My biggest question was whether I could use MLAs to accurately predict if a player was an All-Star for a given season based only upon commonly available statistics. This was my big question because players are only selected from personal opinions, and not statistics. Then, if I can separate these players, I want to know what statistics are the biggest determinants of what makes an All-Star an All-Star.

My personal motivation for applying MLAs to NBA data stems from my love for sports, sports statistics, and my career goals as an aspiring data analyst. I grew up playing sports, and basketball is by far my favorite to play and watch. Just about every night, I will go onto my ESPN app and check all the scores of the basketball games, including the box scores and team stats to get an understanding of what happened in the games. It was because of this fascination for sports statistics that I decided to major in Data Analytics, with eventual hopes of applying what I have learned to actual sports data as a career.

My ultimate goal for this project was to learn how to use different MLAs in Python on data. I made my own Perceptron before this project, so I wanted to apply the other algorithms I learned about in class to data, to see how they work on real-world data. Beyond this being a learning experience, I wanted to produce a classifier that could with some accuracy predict All-Stars. I did not set a specific target accuracy score for my classifier, but I knew I would be happy if my classifier predict 80% of players accurately, and really happy if I saw above 90% accuracy.

The challenges I had for this project were learning the workings of different MLAs in Python, more specifically the ones inside the Sci-Kit Learn package, gathering the data, cleaning the data to provide the best learning for the classifiers, and how to tune the models to get the performance. My approach to address these challenges, was to start with what I knew how to do, and through trial-and-error (lots of error), produce a model that had at least some value.

Through my approach, I was able to successfully implement 3 different classifiers with over 95% accuracy on testing data, with my best model being a SVM with approximately 97% accuracy, followed by an XGBoostClassifier and a Perceptron.

Data Mining Task

The data mining task I sought to achieve was classification. The task consisted of training a classifier model on training data, and then evaluating it on testing data. The goal was to develop the best classifier possible, from the algorithms I learned about in class. I employed this classification task on 3 different MLAs, Perceptron, Support Vector Machines, and ensemble XGBoost. All of the models used the same data for both training and testing, for consistency. The classifiers, given input data, would then predict whether the data represented an All-Star.

The input data all 3 models saw was a NumPy array consisting of numeric values, with each row representing a player. In each row there were 47 different values, each representing a metric. I gathered data from the past 20 seasons (2003-2022), and after cleaning had 6,290 training examples with 520 All-Stars, but this data was given 70/30 split for training and testing. Details on cleaning are discussed later. The data each model was training on consisted of 4,403 examples, with 376 All-Stars. The testing data was composed of the remaining 1,887 players, including 144 All-Stars. The very first training I did used binary (two-class) data on Perceptron. This data had only -1, or 1 for each value, indicating whether that player was above the average for their group. The groupings are discussed later on. After, experimenting with this data, I decided to standardize the data before feeding it into the model again. After decent results and improvements, I experimented by changing the input data to be continuous, rather than binary. Now each value represented the difference between a player's stat, and the mean for their group. This provides more info, as it explains how far above or below average a player was. Again, standardizing the data, I trained Perceptron and got the best results. I tried going even further, having the values be the z scores, but that did not yield better results. The dataset used on the other models, SVM and XGBoost, was the second one described, with values being the difference from the mean standardized, as it yielded the best results with Perceptron. See Figure "Training Example" for an example of what one training example for my models looked like.

The output of the classification problem then, were 3 different classifier models I had learned about in class. I had 1 final model for Perceptron, SVM and XGBoost. For each, I outputted their training and testing accuracy. For the Perceptron and XGBoost algorithms, I was also able to output details regarding the innerworkings of the models. For Perceptron I retrieved the feature weights, which explain the how each feature (player metric in my case) helps to separate the All-Stars. Additionally, with the XGBoost algorithm, I was able to output the decision tree, which shows exactly how an example gets classified. Unfortunately for my best performing model, the SVM, it is not possible to determine feature weights, because the data is transformed to higher dimensions via the kernel trick, with the Radial Basis Function (RBF) kernel method.

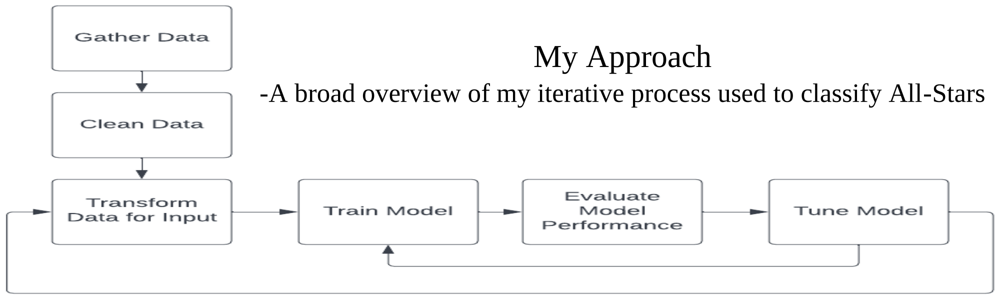
The specific data mining questions I set out to investigate were:

- Are All-Stars separable from non All-Stars based upon commonly available player statistics?
- If they are separable, what are the most impactful metrics (features) that allow players to be separated?
- What MLA can perform the best at separating these groups?

The key challenges for this project include data gathering, cleaning, formatting, classifier training, and classifier hyperparameter tuning. Where should my data come from? What should my metrics my data consist of? How do I clean my data of outliers that will hurt model performance? In what form should I transform my data into for the best model performance? How can I make the best performing model by adjusting the hyperparameters?

Technical Approach

My technical approach to solving this problem, was starting slow, doing what was familiar to me, before branching out into new territory. As mentioned before, I was very unfamiliar with the MLA packages in Python, so I did not really know what would work, and what would not.



My Approach

-A broad overview of my iterative process used to classify All-Stars

Gather Data:

I started with obtaining my dataset. I decided that I wanted to look at the past 20 years of data, so I gathered per game statistics and advanced statistics from basketball-reference.com for each of the past years for each player. I downloaded these as CSVs by year. I then gathered the All-Star game rosters, again as a CSV, so I knew the class labels for the statistics.

Clean Data:

I had to clean the data before I fed it into my model. I loaded my data as Pandas Data Frames and removed unneeded columns, extra rows of players that switched teams, cleaned player names to match that of my All-Star class labels data, and adjusted positions if a player had multiple. I also joined my two CSVs into one dataset. It is here in this step I also had to make important decisions. Because a few of the statistics are expanded to per game, or per 48 minutes, strong outliers can exist. I set thresholds for minimum requirements of certain stats to alleviate these inconsistencies, while making sure to keep every All-Star in my data. For my data, a player had to play at least: 50 total minutes, average of 15 minutes per game for 4 games, and attempt 1 field goal per game. This greatly helped reduce outliers. For example, someone who only attempted 1 shot and made it all season, would have a field goal percentage of 100, an anomaly (50% in general would be amazing, on average). Win Shares per 48 (WS/48) is another statistic I sought to filter for. This statistic determines how helpful a player was in helping the team win, adjusted to 48 minutes, so if a player played only 5 minutes in a couple games, and the team happened to go on a run (outscore the opponent by a wide margin), then their WS/48 would be anomalously high, and I know WS/48 is a great metric for overall player impact.

Transform Data for Input:

Starting out with what I knew about Perceptron, I knew I could train a model based on -1 and 1 values indicating whether a feature was present. I decided my input would be whether a player was better than the average for their group. Because players are selected to be an All-Star by position I grouped them into those 2 categories, frontcourt and backcourt, and because they are selected each year, I also grouped them by year, so in total by position and year. With these groupings, I computed averages for each metric, and set the value to -1 or +1 indicating they were below or above the average. After this, I produced another dataset consisting of stats as the difference to their mean, instead of just -1 or +1 to better illustrate how different they are from each other. I also created a dataset consisting of z scores instead of just the difference to the mean. All the datasets were split into the same training and testing with a 70/30 split, and then standardized with sklearn's Standard Scaler according to the training data.

Train Model:

Starting with sklearn Perceptron, I trained it on the binary inputs training data unstandardized and standardized. I then trained another Perceptron on the second and third datasets in similar fashions. I found that my second dataset provided the best results, so it is with this data, I decided to train sklearn SVMs and XGBoost's (from xgboost library).

Evaluate Model Performance:

Here I determine how well my model performed on both the training data and the testing data. While both are important, I am careful to not overfit to the training data, because how well the model generalizes to unseen data is more important. The main goal is to maximize my testing accuracy, even if I have to sacrifice training accuracy.

Tune Model:

I attempted to tune the model for the best performance possible. I utilized the GridSearchCV algorithm to test different model hyperparameters on each model. For Perceptron I experimented with learning rates. For SVM I tested different C values, or how much I want to avoid wrongly classifying training examples. For XGBoost, I tested number of estimators, learning rate, and maximum depth of the booster trees.

Evaluation Methodology

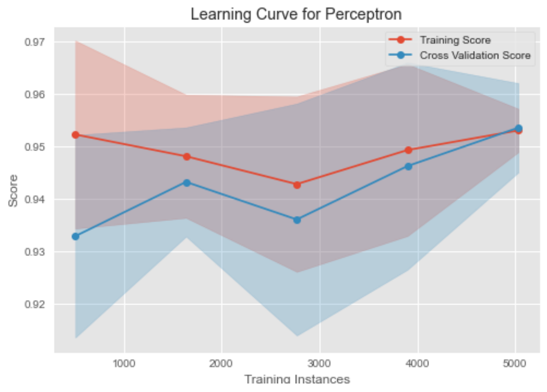
My dataset consists of 47 different metrics gathered from basketball-reference.com. I built my dataset by combining the "per game" and "advanced" player statistics for each year. The per game dataset contained 23 unique features, and the advanced dataset had 20 unique features, while both shared 4 features. I joined separate datasets together by tracking their names and the year of the data. The only challenges from this dataset came in gathering and cleaning it. I could not find great functionality for downloading the specific datasets I wanted, so I tediously exported each one manually as a CSV from the website for each year. Additionally, I had to play around with different minimum thresholds mentioned above to exclude outliers.

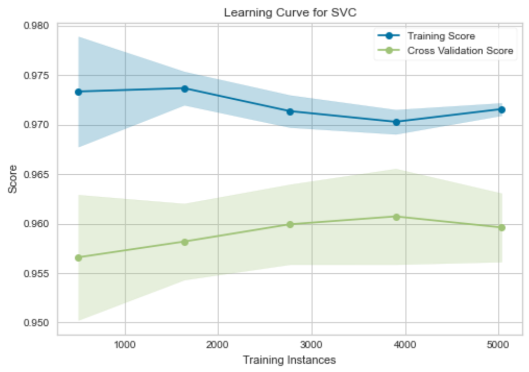
The metric I used to evaluate the success of my classifier models, and whether or not I could identify All-Stars, was the testing accuracy of my models. If my models could generalize to my testing split of data, with high accuracy, then I could answer positively, that, yes, MLAs can identify NBA All-Stars. This testing accuracy also served to tell me, which one of the MLAs I explored performed the best, knowing that the data was separable. In terms of real-world application, this testing accuracy is truly what we care about. We do not care if a model can memorize data is has already seen, we want the model to perform well on data it has not seen. For my last question, regarding which metrics are most important in determining All-Stars, the features weights, and importances, where applicable, would be the numbers I use to see which metric in my dataset has the biggest impact.

Results and Discussion

After following, the outlined steps above, I have produced 3 different classifier models, a Perceptron, a Support Vector Machine, and an XGBoostClassifier, which are all able to classify All-Stars from non All-Stars from the past 20 seasons at over 95% testing accuracy. I first tested my data on Perceptron, and this told me that the data was separable, as I was achieving over 90% testing accuracy for all of my datasets. I tested all 3 versions of data formats I created, standardized and unstandardized, on Perceptron, and achieved the best test accuracy with my stat minus their mean for the group, achieving 96.18% testing accuracy.

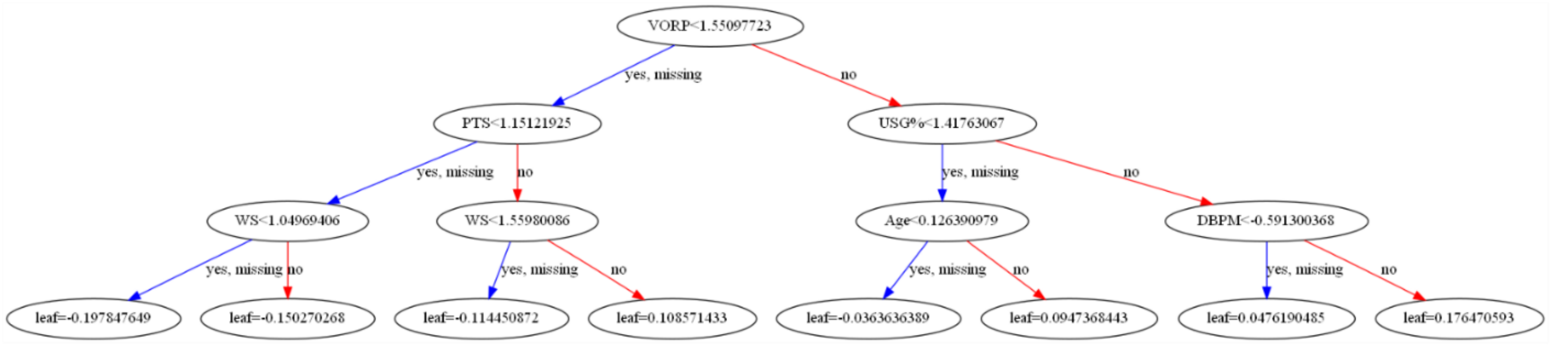
After knowing, for sure, my data was separable, I went on to see how well other MLAs could classify my data. With my Perceptron already achieving such as high accuracy, I was not expecting major improvements. I next tested an SVM, using the same data. This SVM performed even better than my Perceptron, albeit only a minor improvement, with a testing accuracy of 97.09%.





My third Algorithm implemented was XGBoostClassifier. This classifier performed well, better than my Perceptron, but worse than my SVM at 96.40% testing accuracy with a depth of 3. I did develop an XGBoostClassifier that performed at 96.45% but that classifier had a depth of 5, and using Occam's Razor, I decided to disregard it, as it had 100% training accuracy, and a very complex decision tree. Below is my less accurate (barely), but simpler Decision Tree.

Decision Tree from my 96.40% Accurate XGBoost



-negative leaves = non All-Star
-positive leaves = All-Star

Now that I know I can classify the data, and that the SVM classifier performed the best, I want to answer, what features had the most importance. Unfortunately for my best classifier, the

SVM, I cannot determine this question. My SVM uses the RBF kernel trick, which expands dimensions, and therefore the weight coefficients have no meaning in my data's original dimension space. I was, however, able to determine weights for my XGBoost and Perceptron. For my XGBoost, the most important features in order using the gain, not weight, were VORP, FG, DWS, PER, and PTS (Figure "XGBoost Feature Importance"). These features make sense to me for the most part, with my biggest surprise coming in as DWS. VORP, value or replacement player, is a numerical score for how many points this player contributes over a replacement (average) player, so this being a major separator makes sense, as this essentially says how much better than an average player this player is, and this received an importance score 50% higher than the next best feature, FG. PER is another overall metric, that tries to sum up a player's accomplishments per minute. DWS is surprising to me because it stands for Defensive Win Shares, which says how much a player helps their team win through their defense. This surprises me, as it was my assumptions that thought fans valued offensive skills much more than defense, when it comes to All-Star voting. My assumptions do have some truth though as 4 of the 5 most important features relate to offense VORP, FG, PTS, and PER, while DWS is the only metric that only accounts for defensive abilities. I used the 'gain' rankings, as opposed to the 'weight' rankings which can be seen in the actual decision tree. (Refer to https://www.basketball-reference.com/about/glossary.html for explanations of the stats I used. See "Figure Features I Used" for the stats I used.)

For my Perceptron, the 5 highest weights were for WS, VORP, PTS, OWS, and FTA. There are 2 overlapping stats from the gain ranked features from XGBoost, VORP and PTS. This the evidence towards my belief that offensive skills play a heavy role in determining All-Stars, as every single one of these stats related directly to offense.

The methods that worked best for my models was taking the stats and subtracting the mean for their groups, before standardizing it and training the models. Overall, all 3 models I trained were able to perform very well on the data.

What did not work as planned was my tuning of model hyperparameters. through various testing of Sci-Kit Learn's GridSearchCV algorithm, I was unable to see impactful improvements in my models' performance. The models did perform well before tuning the parameters, I just hoped to see improvements, when I did experiment.

Lessons Learned

Through this whole project, I learned a lot! I learned how to implement a multitude of Machine Learning Algorithms, and how great the packages sklearn and xgboost are. Most importantly, I learned that using Machine Learning Algorithms are not really such a big mystery, like I thought before. The algorithms don't really perform any magic, they just can interpret a lot of data, very fast and in ways humans cannot. In hindsight, I would have looked for more features to add to my algorithm and added more data to train on from even more years. I also would try to experiment with hyperparameter tuning more to get better models. I definitely plan on experimenting with Machine Learning Algorithms more, now that I see they're not so scary!

Acknowledgements

Where I sourced my statistics from:

https://www.basketball-reference.com/leagues/NBA_2022_advanced.html

https://www.basketball-reference.com/leagues/NBA_2022_per_game.html

How the NBA All-Stars are selected:

https://www.nba.com/news/all-star-draft-explainer

Where I found All-Star Game Rosters:

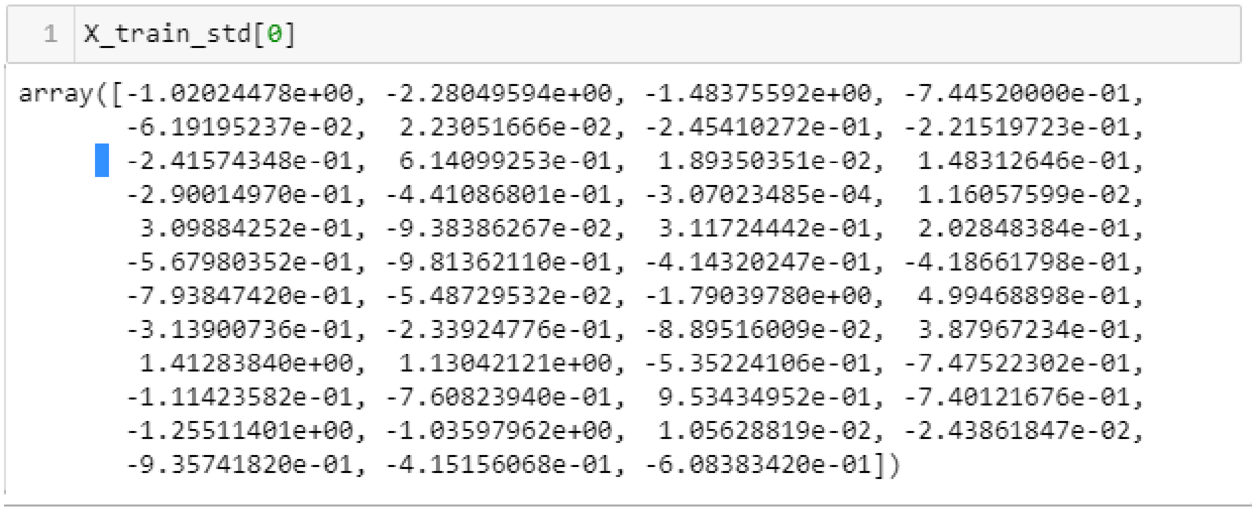
https://basketball.realgm.com/nba/allstar/game/rosters/2022

A function I used to plot feature importances for Perceptron

https://stackoverflow.com/questions/57082102/feature-importance-using-svms-coef-function

Figures

Training Example

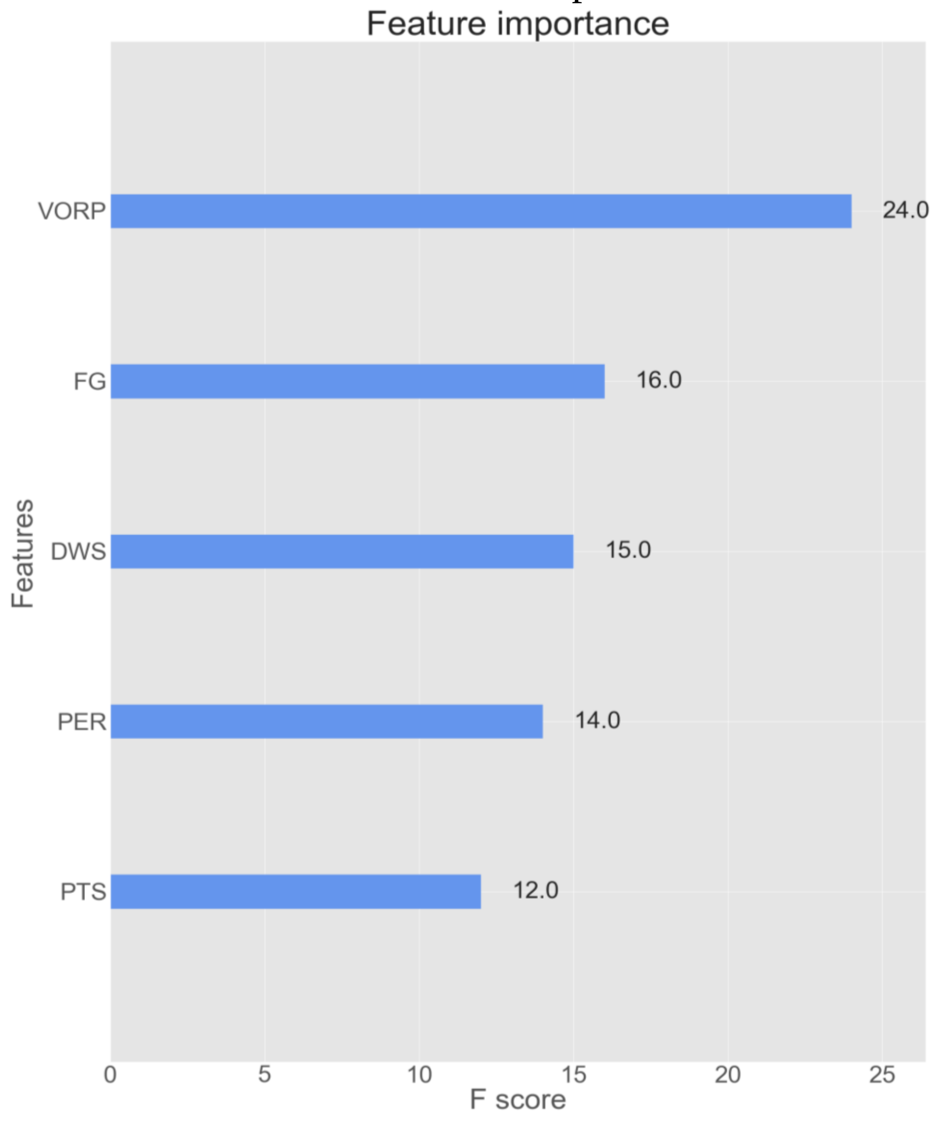


```
X_train_std[0]
```

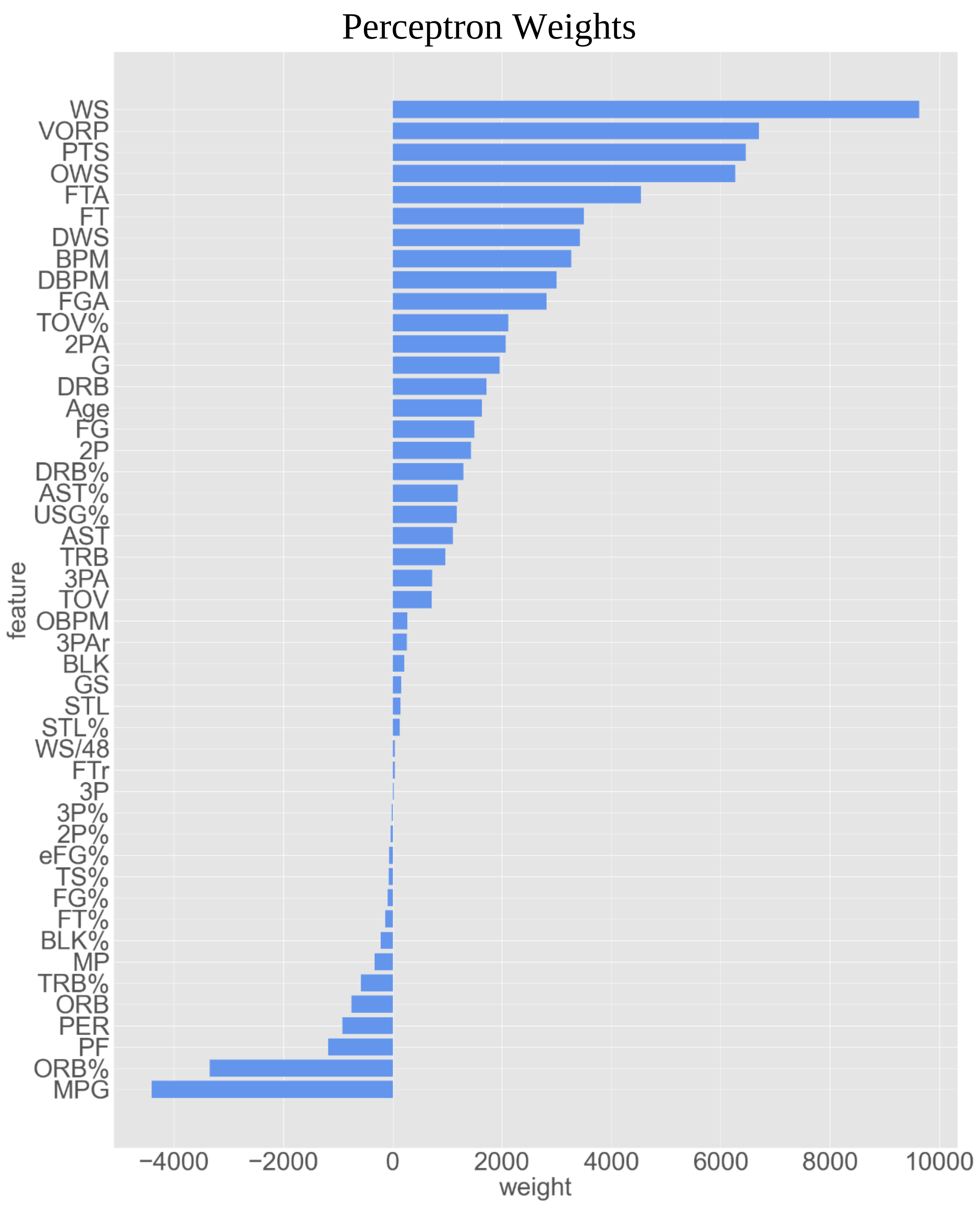
```
array([-1.02024478e+00, -2.28049594e+00, -1.48375592e+00, -7.44520000e-01,
       -6.19195237e-02,  2.23051666e-02, -2.45410272e-01, -2.21519723e-01,
       -2.41574348e-01,  6.14099253e-01,  1.89350351e-02,  1.48312646e-01,
       -2.90014970e-01, -4.41086801e-01, -3.07023485e-04,  1.16057599e-02,
        3.09884252e-01, -9.38386267e-02,  3.11724442e-01,  2.02848384e-01,
       -5.67980352e-01, -9.81362110e-01, -4.14320247e-01, -4.18661798e-01,
       -7.93847420e-01, -5.48729532e-02, -1.79039780e+00,  4.99468898e-01,
       -3.13900736e-01, -2.33924776e-01, -8.89516009e-02,  3.87967234e-01,
        1.41283840e+00,  1.13042121e+00, -5.35224106e-01, -7.47522302e-01,
       -1.11423582e-01, -7.60823940e-01,  9.53434952e-01, -7.40121676e-01,
       -1.25511401e+00, -1.03597962e+00,  1.05628819e-02, -2.43861847e-02,
       -9.35741820e-01, -4.15156068e-01, -6.08383420e-01])
```

-this is an example of one training example my MLAs were fitted on. This was the statistic minus the mean for a players group(frontcourt/backcourt, year), then scaled after a Standard Scaler was fitted on the training data.

XGBoost Feature Importance

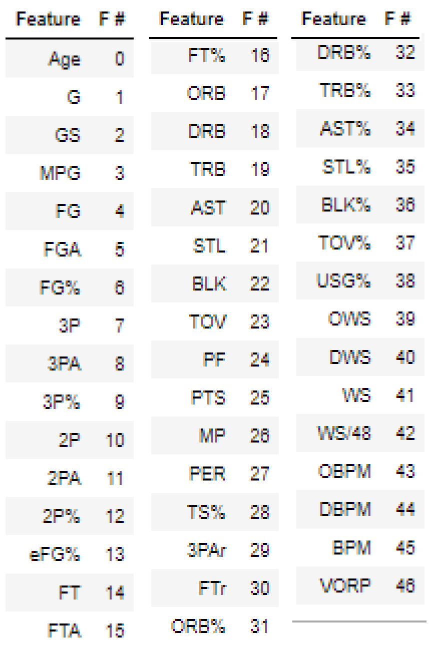


-this chart contains the top 5 most important features in my dataset that determine an All-Star vs one that's not. Something very important to note here is that these are importance ranked according to gain, and not weight.



-this chart plots the feature weight for every single metric. The features on the higher end of the spectrum are more telling that that player is an All-Star, and the features on the lower side have less of tell-tale sign the player is an All-Star, all else being equal.

Features I used



Feature	F #
Age	0
G	1
GS	2
MPG	3
FG	4
FGA	5
FG%	6
3P	7
3PA	8
3P%	9
2P	10
2PA	11
2P%	12
eFG%	13
FT	14
FTA	15
FT%	16
ORB	17
DRB	18
TRB	19
AST	20
STL	21
BLK	22
TOV	23
PF	24
PTS	25
MP	26
PER	27
TS%	28
3PAr	29
FTr	30
ORB%	31
DRB%	32
TRB%	33
AST%	34
STL%	35
BLK%	36
TOV%	37
USG%	38
OWS	39
DWS	40
WS	41
WS/48	42
OBPM	43
DBPM	44
BPM	45
VORP	46

Refer to https://www.basketball-reference.com/about/glossary.html for explanations of the stats