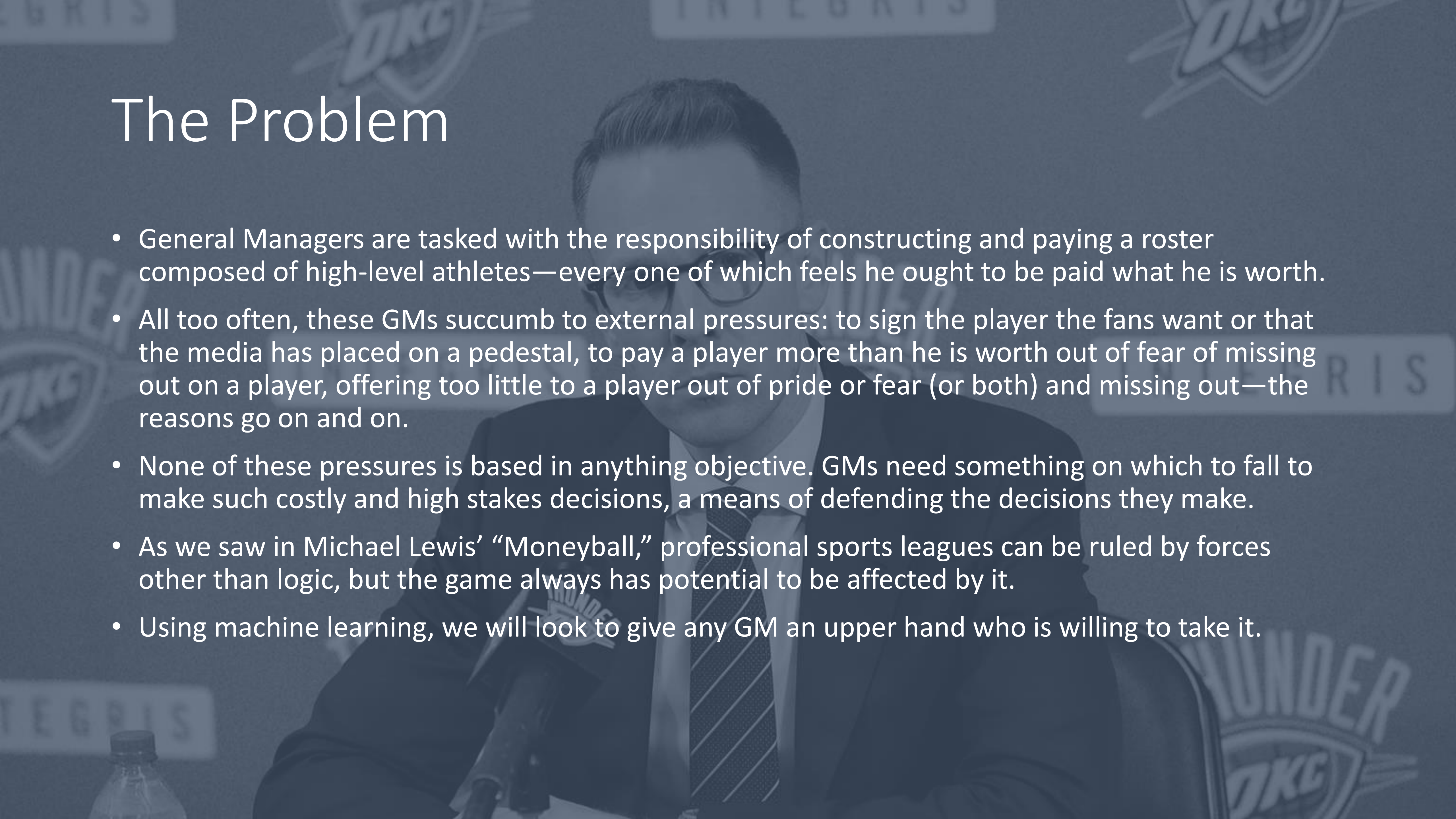
NBA Player Value

Using Machine Learning to Better Run an NBA Franchise: Milestone 2



The Problem

- General Managers are tasked with the responsibility of constructing and paying a roster composed of high-level athletes—every one of which feels he ought to be paid what he is worth.
- All too often, these GMs succumb to external pressures: to sign the player the fans want or that the media has placed on a pedestal, to pay a player more than he is worth out of fear of missing out on a player, offering too little to a player out of pride or fear (or both) and missing out—the reasons go on and on.
- None of these pressures is based in anything objective. GMs need something on which to fall to make such costly and high stakes decisions, a means of defending the decisions they make.
- As we saw in Michael Lewis' "Moneyball," professional sports leagues can be ruled by forces other than logic, but the game always has potential to be affected by it.
- Using machine learning, we will look to give any GM an upper hand who is willing to take it.

Identify Successful Teams

Determine what makes them successful

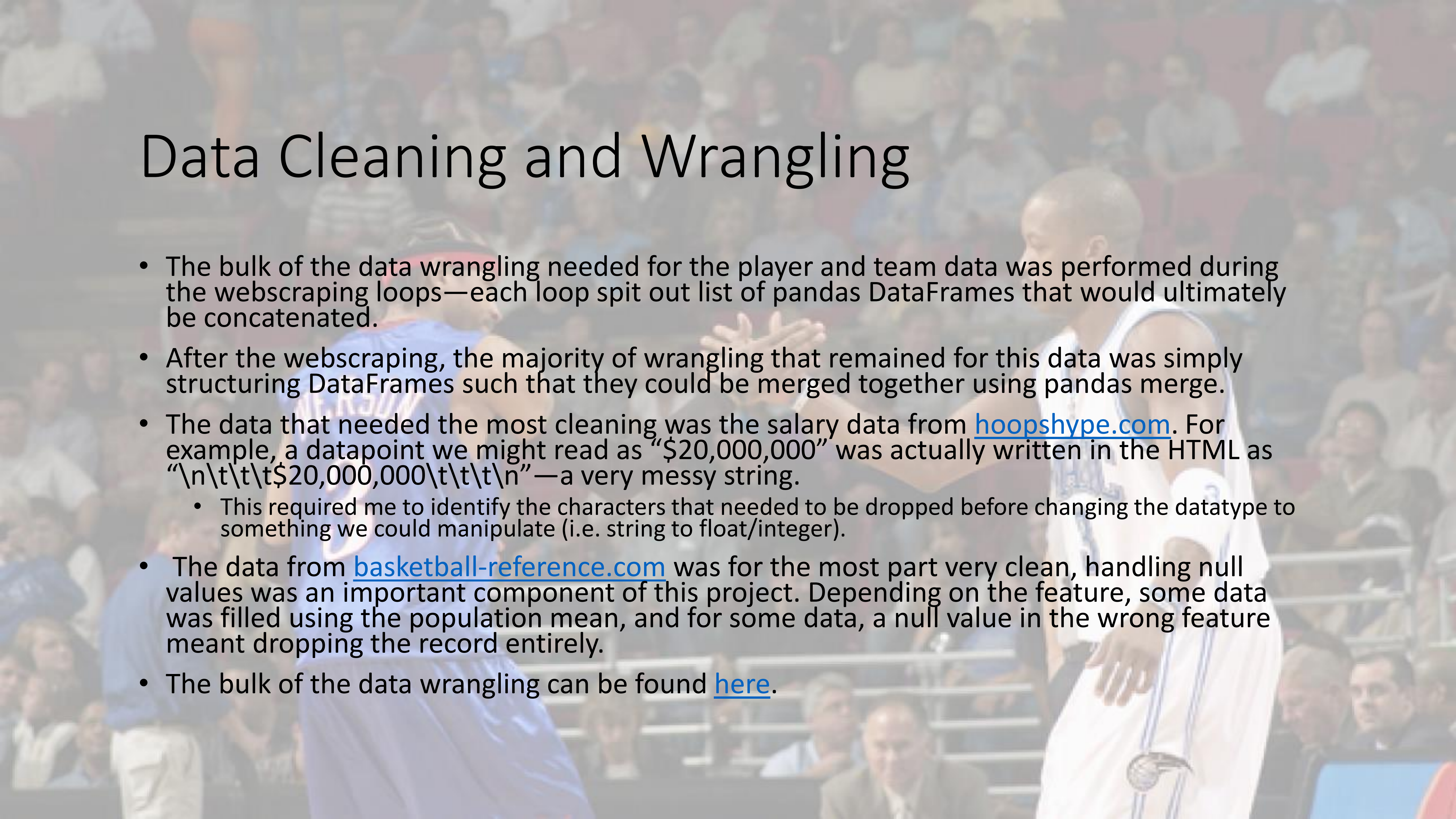
Determine True Player Value

Make Recommendations

Initial Objectives

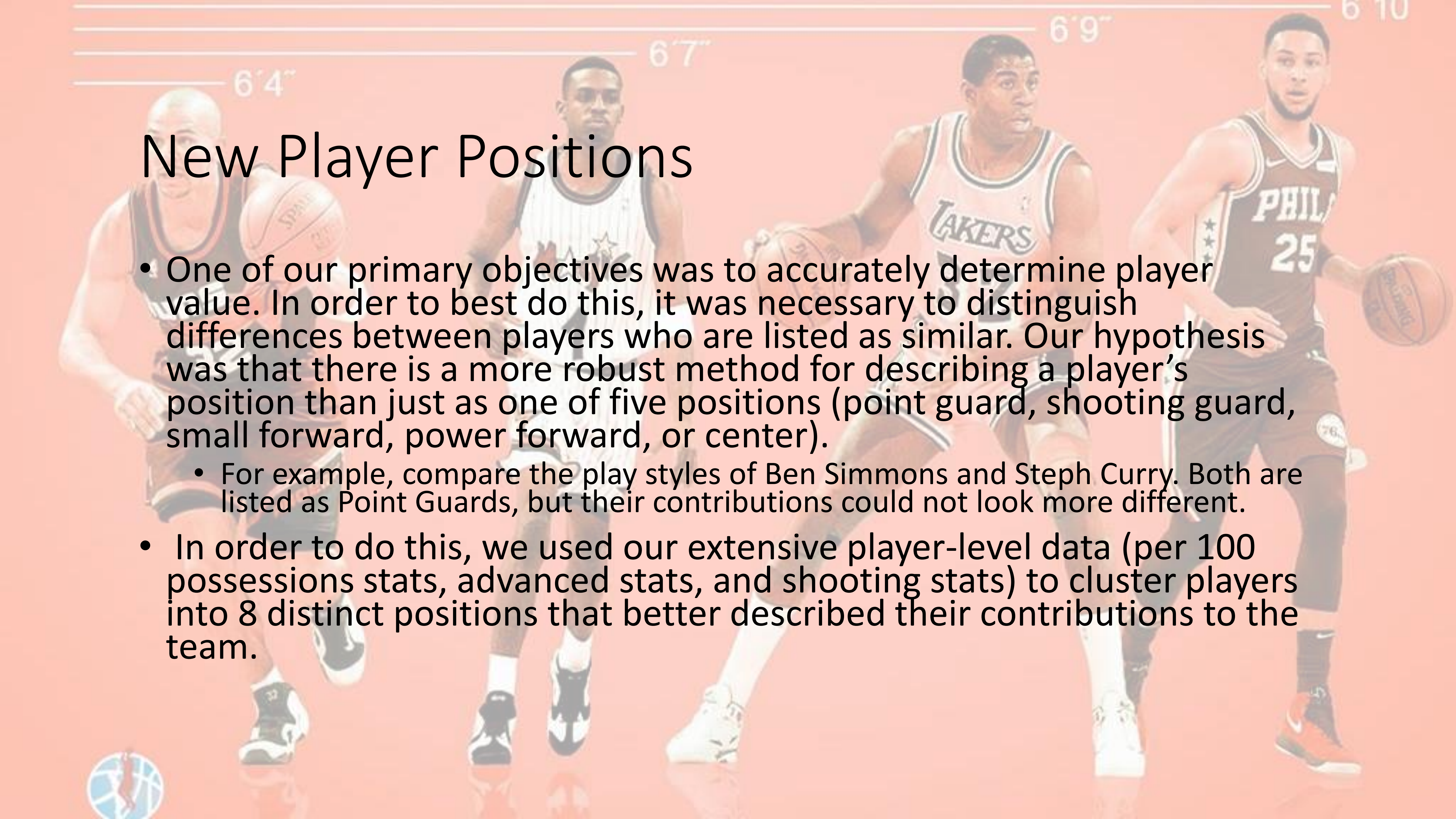
The Data

- The data was acquired via webscraping a few different websites using the BeautifulSoup package.
- Data was drawn across multiple websites and many different urls within each website.
- Player data was a composite dataset of "per 100 possessions" stats, advanced stats, and shooting stats from basketball-reference.com, as well as salary stats from hoopshype.com.
- Team data was also a composite dataset of win-loss data and advanced statistics pulled from basketball-reference.com as well.
- All webscraping code can be found here.



Data Cleaning and Wrangling

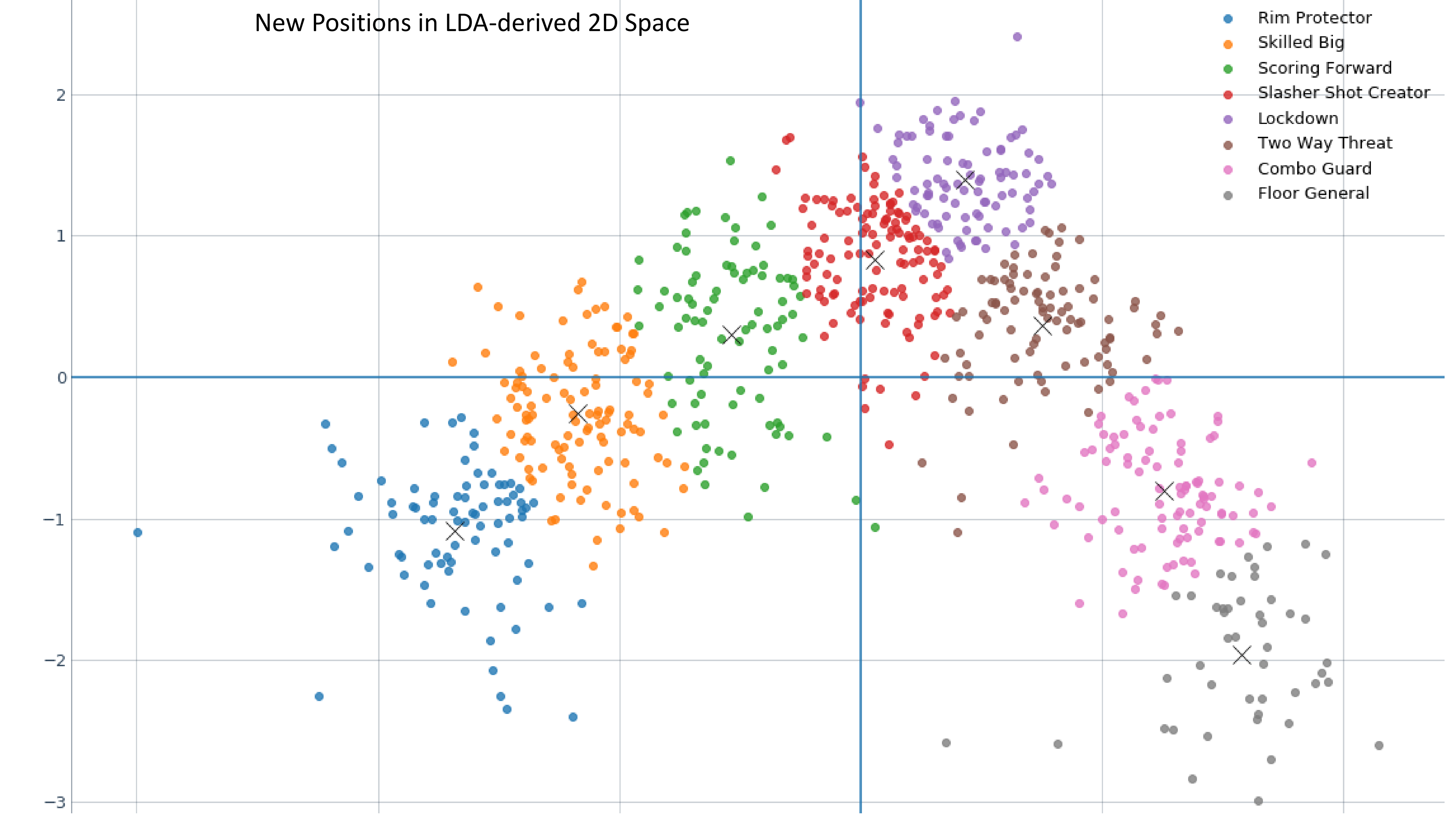
- The bulk of the data wrangling needed for the player and team data was performed during the webscraping loops—each loop spit out list of pandas DataFrames that would ultimately be concatenated.
- After the webscraping, the majority of wrangling that remained for this data was simply structuring DataFrames such that they could be merged together using pandas merge.
- The data that needed the most cleaning was the salary data from hoopshype.com. For example, a datapoint we might read as “$20,000,000” was actually written in the HTML as “\n\t\t\t$20,000,000\t\t\t\n”—a very messy string.
  - This required me to identify the characters that needed to be dropped before changing the datatype to something we could manipulate (i.e. string to float/integer).
- The data from basketball-reference.com was for the most part very clean, handling null values was an important component of this project. Depending on the feature, some data was filled using the population mean, and for some data, a null value in the wrong feature meant dropping the record entirely.
- The bulk of the data wrangling can be found here.



New Player Positions

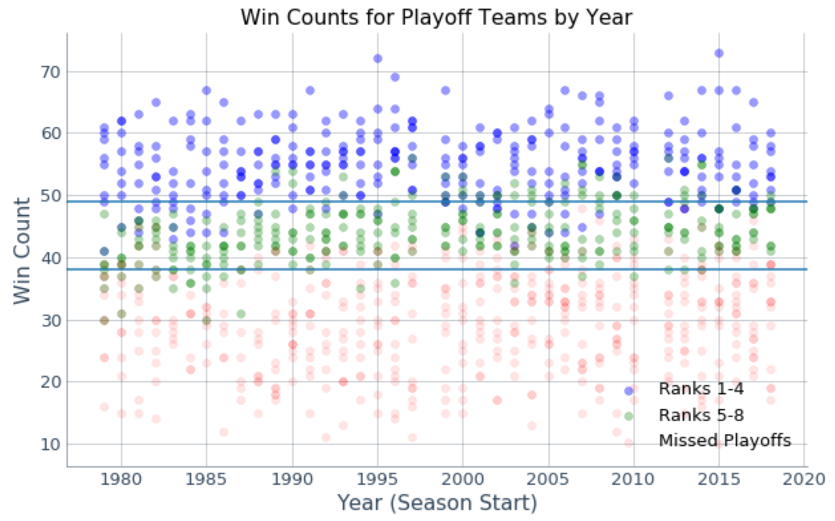
- One of our primary objectives was to accurately determine player value. In order to best do this, it was necessary to distinguish differences between players who are listed as similar. Our hypothesis was that there is a more robust method for describing a player's position than just as one of five positions (point guard, shooting guard, small forward, power forward, or center).
  - For example, compare the play styles of Ben Simmons and Steph Curry. Both are listed as Point Guards, but their contributions could not look more different.
- In order to do this, we used our extensive player-level data (per 100 possessions stats, advanced stats, and shooting stats) to cluster players into 8 distinct positions that better described their contributions to the team.

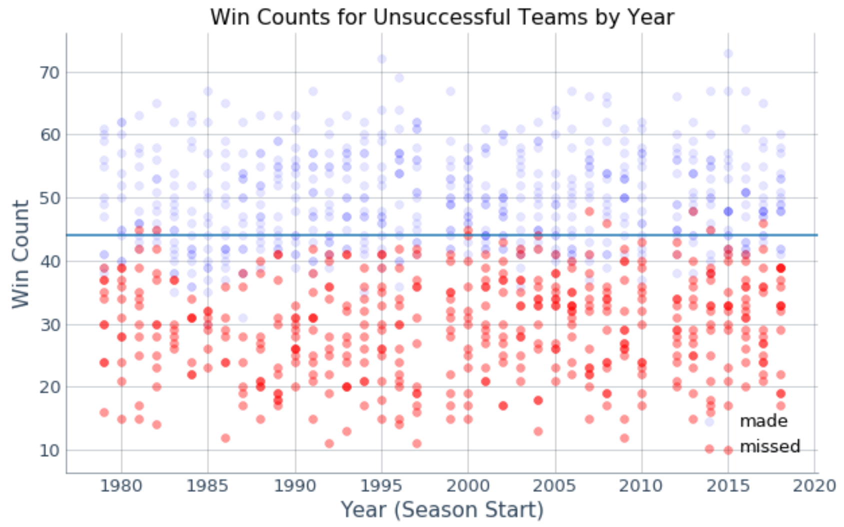
New Positions in LDA-derived 2D Space



Dimensionality Reduction

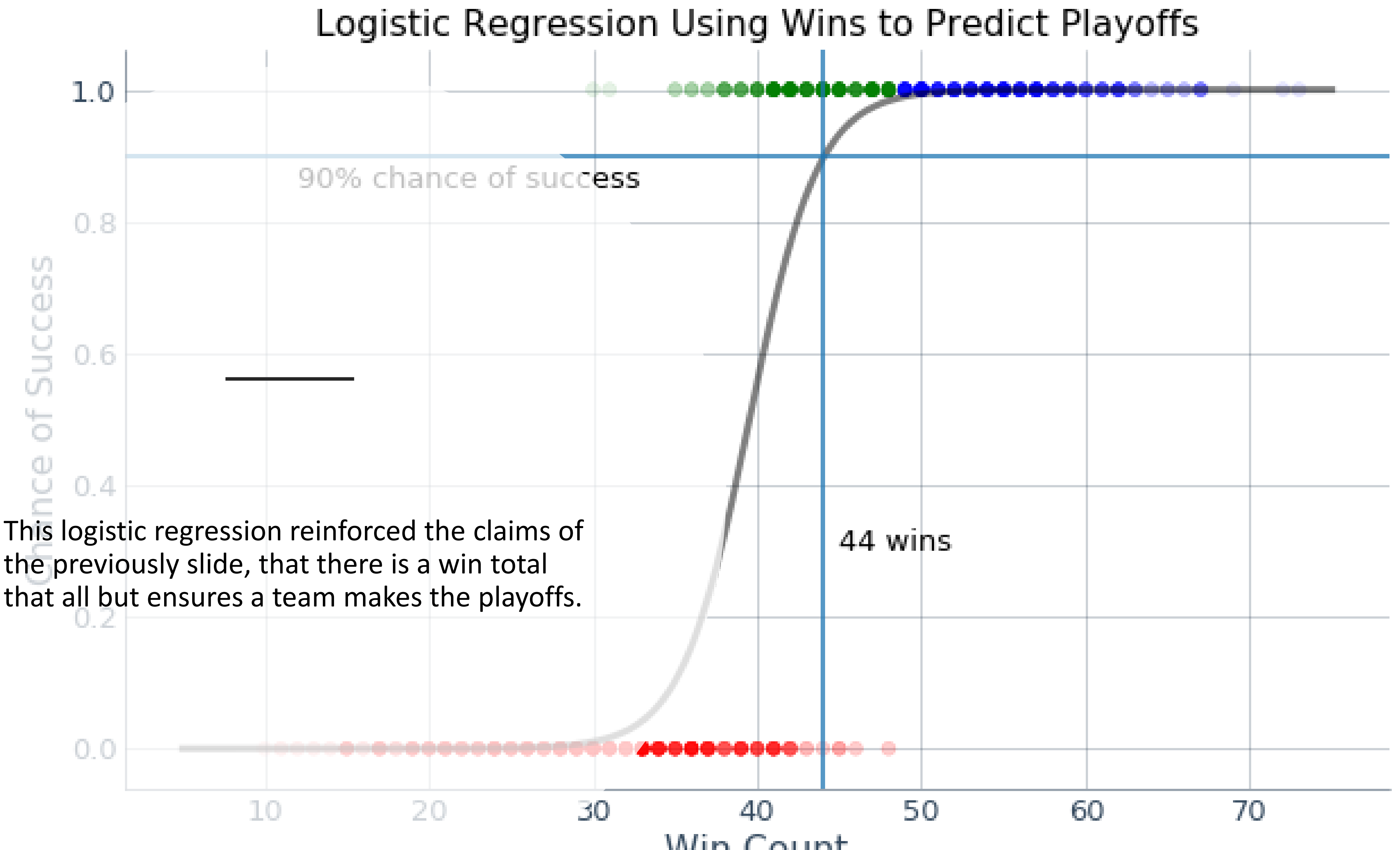
- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- K-means clustering
- Once new labels were merged with our player-level data, we were able to add another layer to our EDA and inferential statistics as we could then consider a player's cluster when asking questions of value.





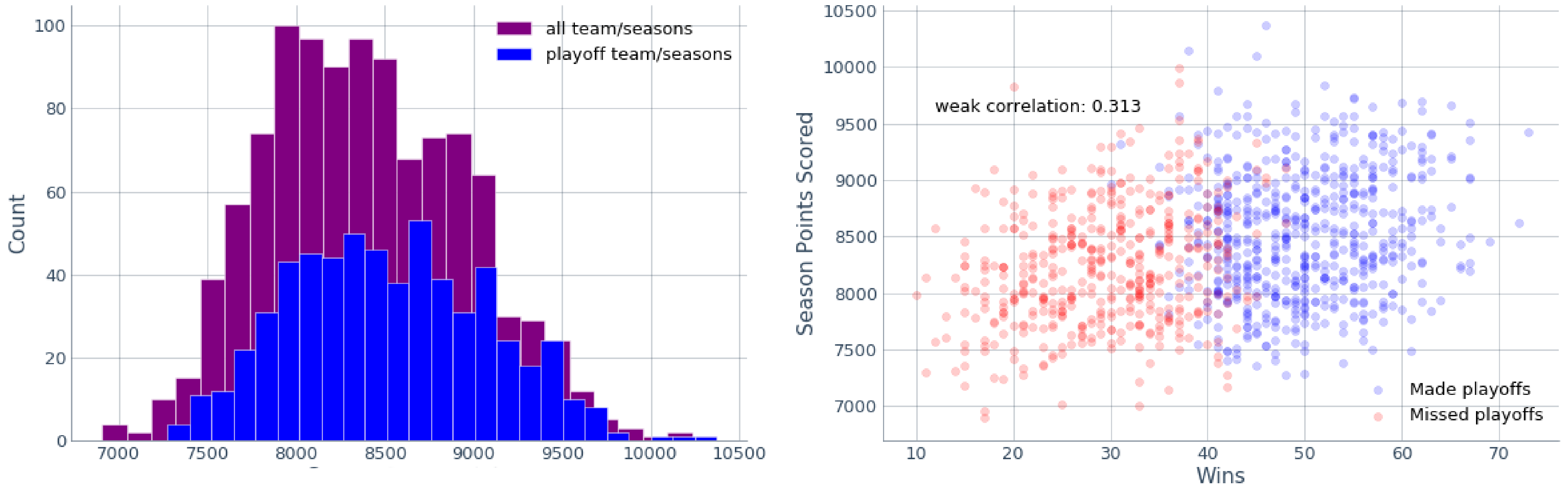
Exploratory Data Analysis

- On the team-level, we wanted to identify success, so that required of us that we define our terms.
  - “Success” for the scope of this project equated to making playoffs, or even better, making the playoffs with a good seeding (1st through 4th).
- Left are two similar plots that show roughly how many wins generated successful season.
  - At 38 wins, a team has a good chance of making playoffs, but at 44 wins, it’s highly unlikely that a team would miss playoffs.
  - At 49 wins, a team will very likely have a good playoff seed.



This logistic regression reinforced the claims of the previously slide, that there is a win total that all but ensures a team makes the playoffs.

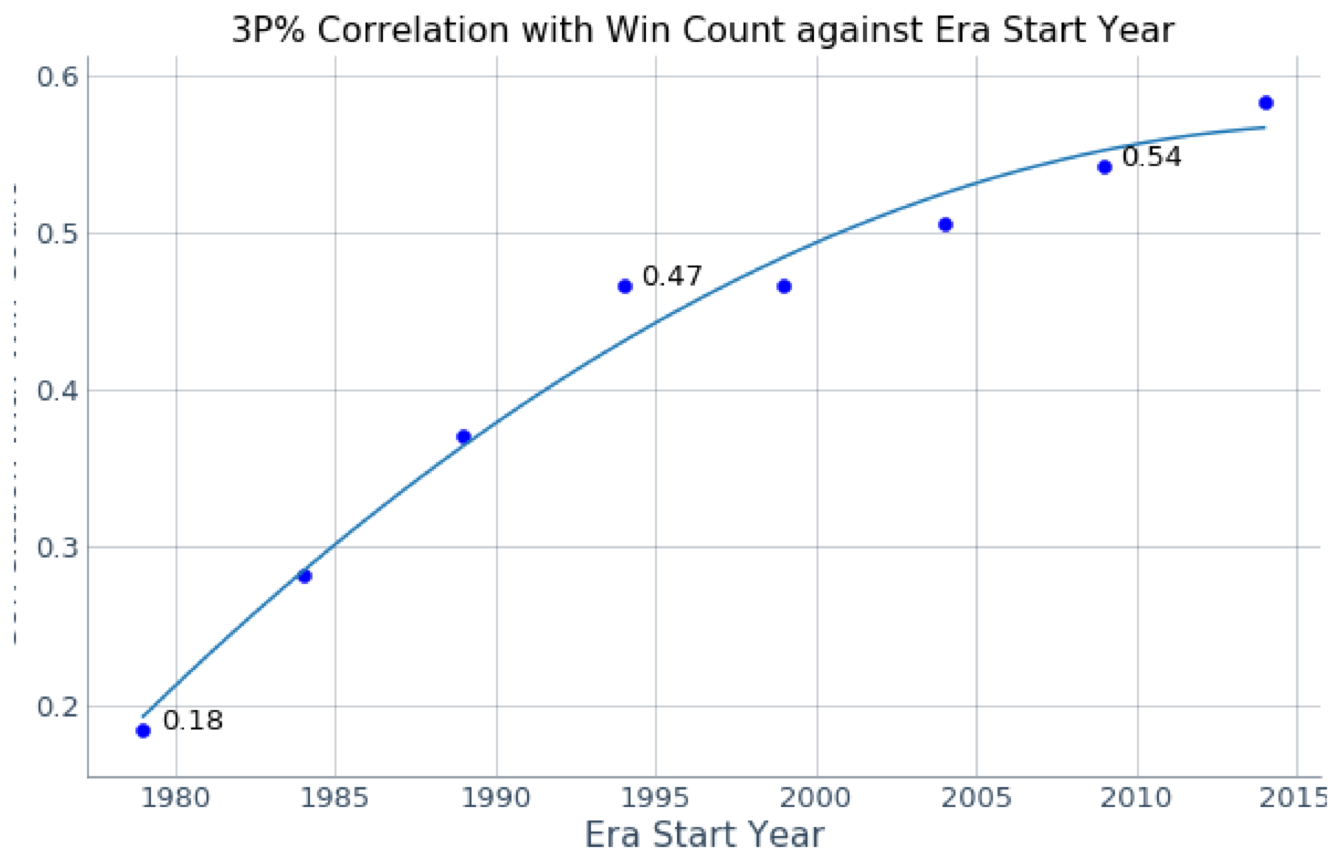
Point Totals for Playoff Team/Seasons

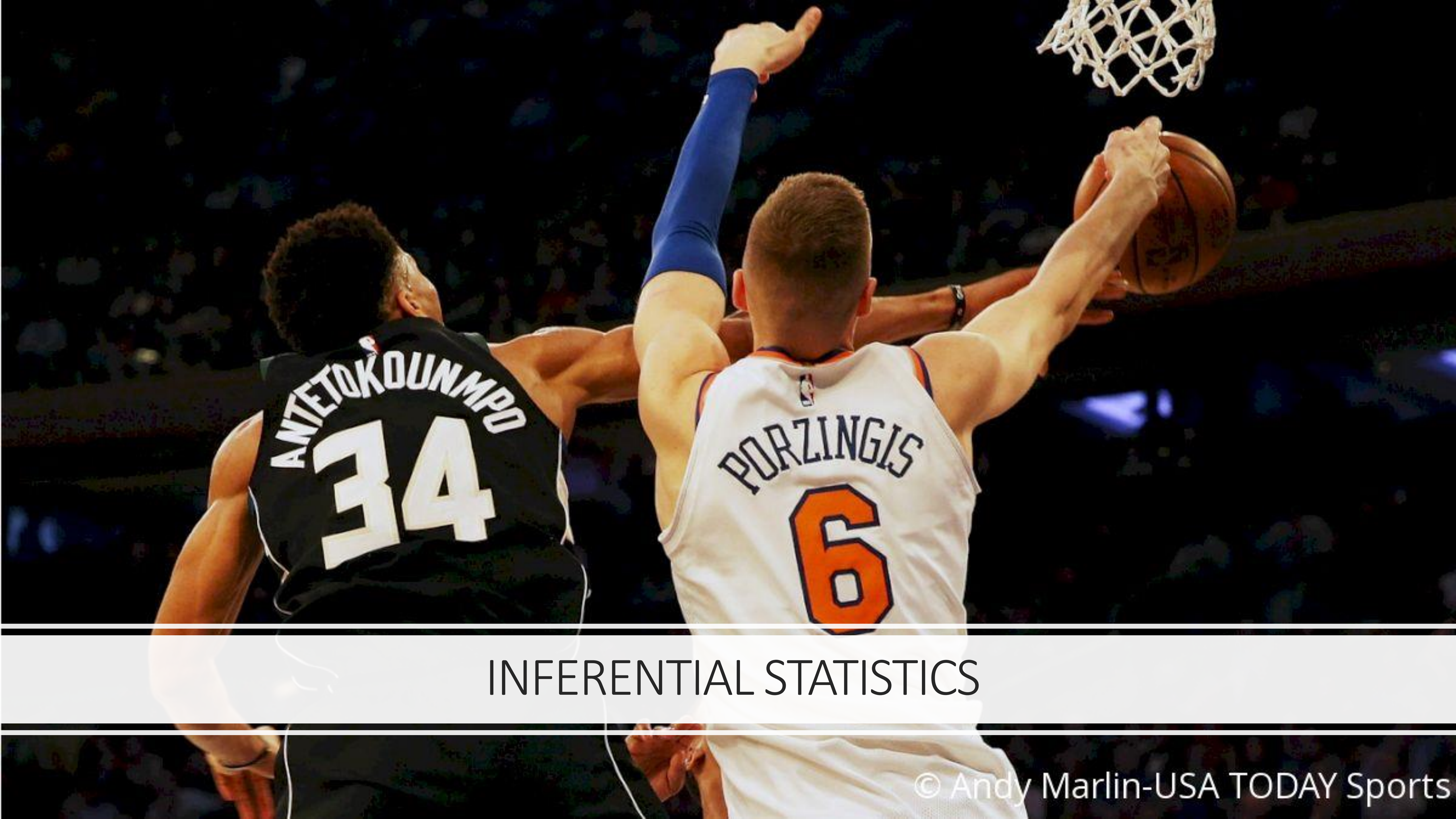


Team-level Win Correlates

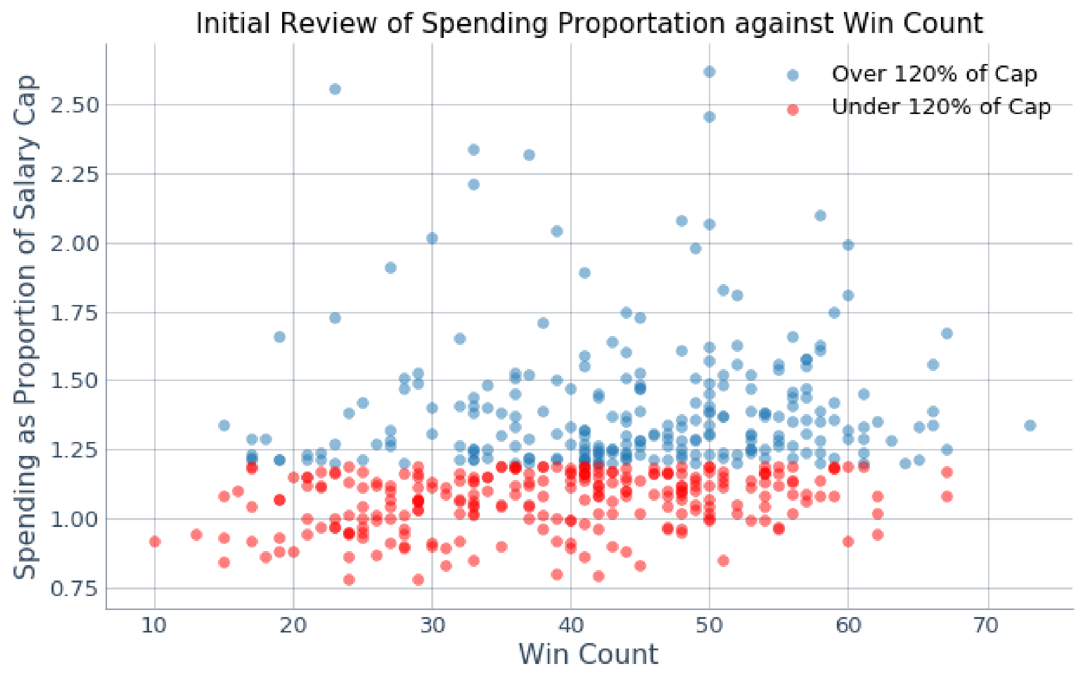
- The next step was to look for correlates of win count. “What did a team that won a bunch of games do well that season?”
- On first pass, we checked the statistic of “points scored”. This, interestingly had a very weak correlation with win count—just because a team is scoring a lot of points, doesn’t mean they’re winning games. Same could be said with the statistics of “points against”.

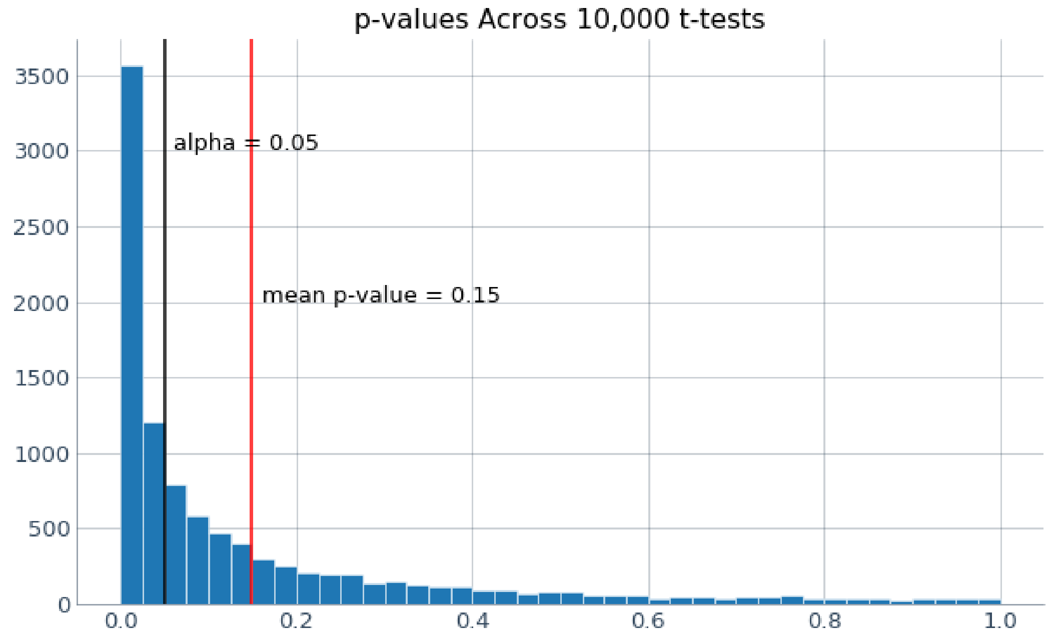
- Where we did see our most notable strong correlation was with the feature “3P%”, or 3-point percentage. Teams that were more accurate from the 3-point line won more games.
  - When we first looked at this correlate, we saw a coefficient of 0.18, but this included team data dating back to 1979.
  - When we limited our data to just team-seasons dating back 5 years (2014-2019) instead, we saw a correlation of 0.58.





INFERENTIAL STATISTICS





Spend to Win?

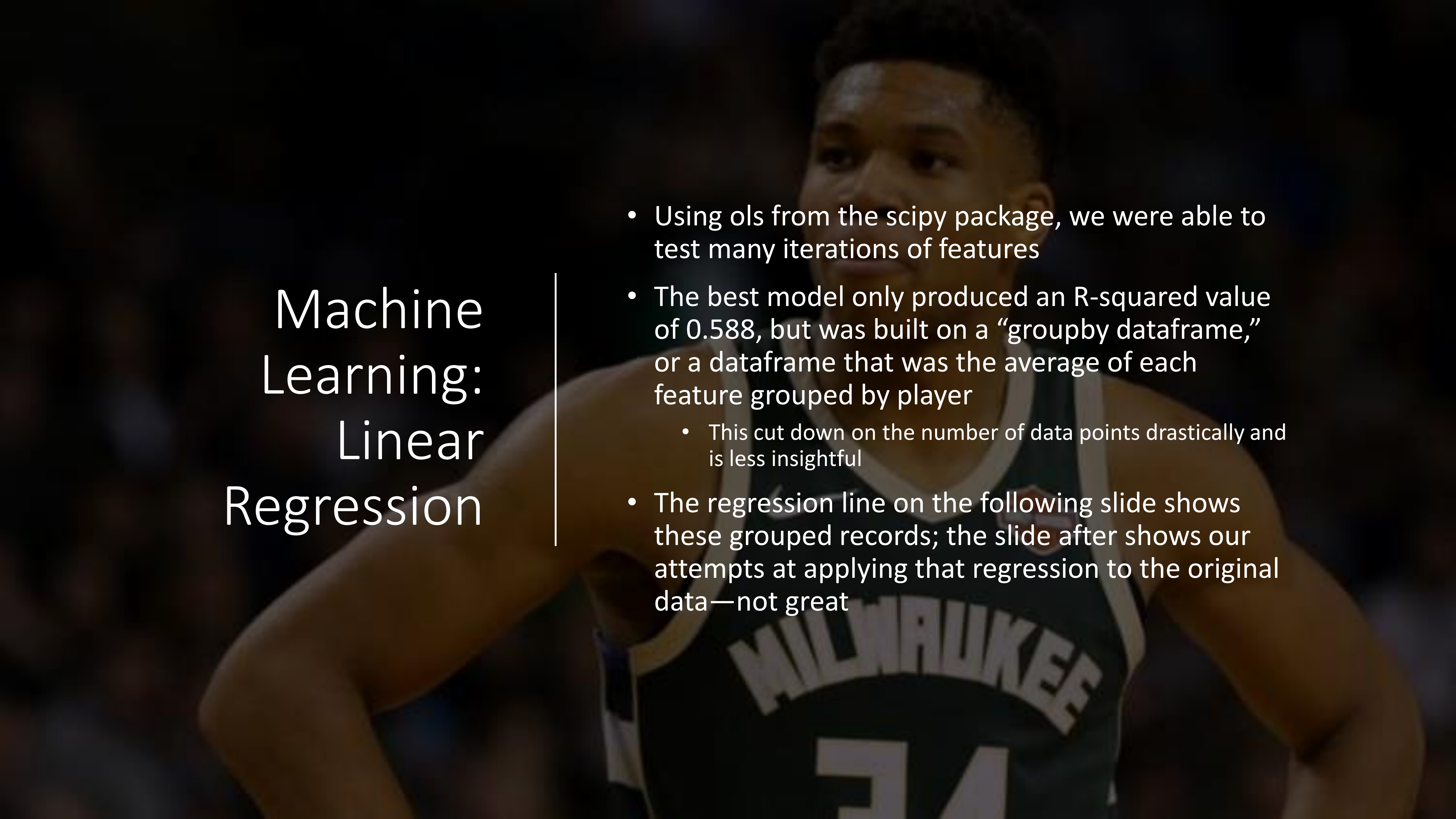
- With our team-level inferential statistics, two sample t-tests of independence showed us that money doesn't buy wins--at least not at a statistically significant level. The differences in mean Win count for teams paying over 120% of the salary cap for any given year, and teams paying under 120% of the salary cap for any given year varied greatly--44 and 38, respectively.
- However, when we sampled from these groups and ran a few t-tests (read: ten thousand t-tests), we saw that that this difference was not in fact significant at even a 0.10 confidence. The resultant p-values were plotted in the histogram (bottom).

Inferential Statistics cont.

The following slide demonstrates how python allowed us to scan for statistically significant differences in 6 different features across 8 subsets of the data, using 10,000 t-tests each time (48,000 in total) in about 2.5 minutes.

- From this scan, we learned the following:
  - the "Lockdown" cluster was flagged for statistically lower mean Player Efficiency Rating (PER)
  - "Rim Protectors" were flagged for having statistically lower mean Usage Rate (USG%)
  - "Combo Guards" were conversely flagged for statistically higher mean Usage Rate
  - "Rim Protectors" were the only cluster with a significant difference (lower) in Points Scored (PTS)

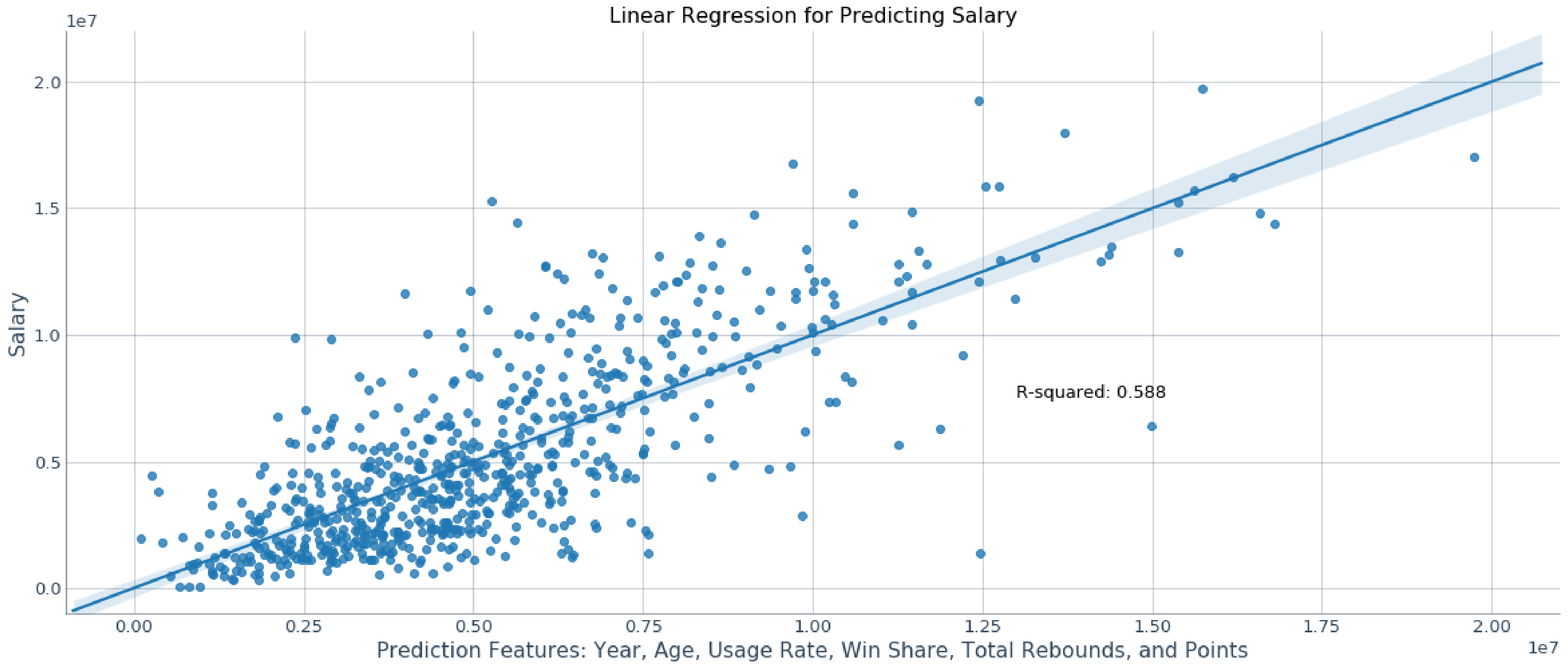
```
clusterlist = players.cluster.unique()
value_stats = ['PER', 'WS', 'BPM', 'VORP', 'USG%', 'PTS']
confidence = 0.05 * 2

for h in value_stats:

    pop_mean = np.mean(players[h])

    for i in tqdm(clusterlist):

        players_T_list = []
        players_p_list = []
        cluster = list(players[players.cluster == i][h])

        for j in range(10000):

            cluster_samp = sample(cluster, 30)
            T, p = stats.ttest_1samp(cluster_samp, pop_mean)
            players_T_list.append(T)
            players_p_list.append(p)

        T = np.mean(players_T_list)
        p = np.mean(players_p_list)

        if p <= confidence:
            print('Cluster: {};\tValue Stat: {}'.format(i, h))
            print('Sample mean {}:'.format(h), round(np.mean(cluster_samp),
                                          2), 'Population mean {}:'.format(h), round(pop_mean, 2))
            print('Test statistic (T):', round(T, 2), 'p-value:', round(p, 3))
            print('Reject the null. Significant difference in mean {} between population and the {} cluster.'.format(h, i))
            if T < 0:
                print('For the {} cluster, mean {} is lower than the league average.'.format(i, h))
            else:
                print('For the {} cluster, mean {} is above the league average.'.format(i, h))
```



Machine Learning: Linear Regression

- Using ols from the scipy package, we were able to test many iterations of features
- The best model only produced an R-squared value of 0.588, but was built on a “groupby dataframe,” or a dataframe that was the average of each feature grouped by player
  - This cut down on the number of data points drastically and is less insightful
- The regression line on the following slide shows these grouped records; the slide after shows our attempts at applying that regression to the original data—not great

Groupby-DataFrame

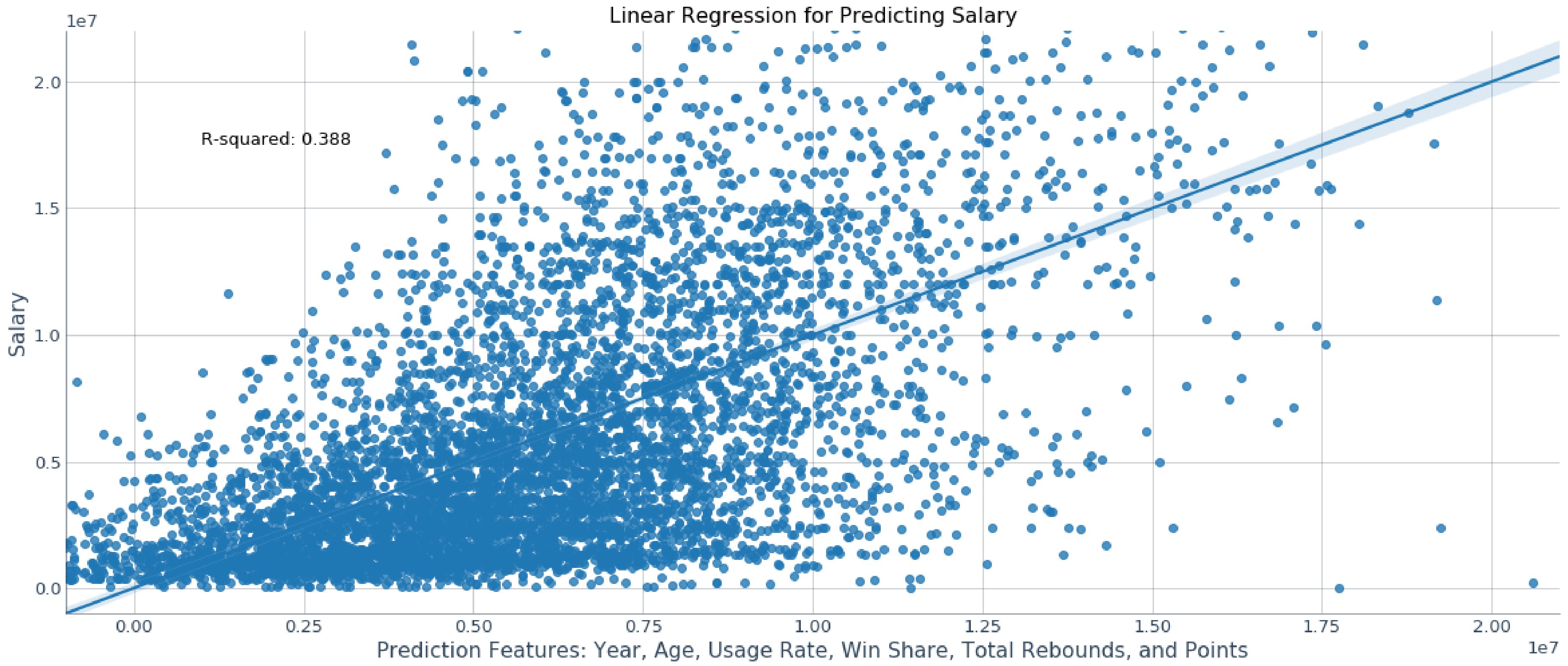


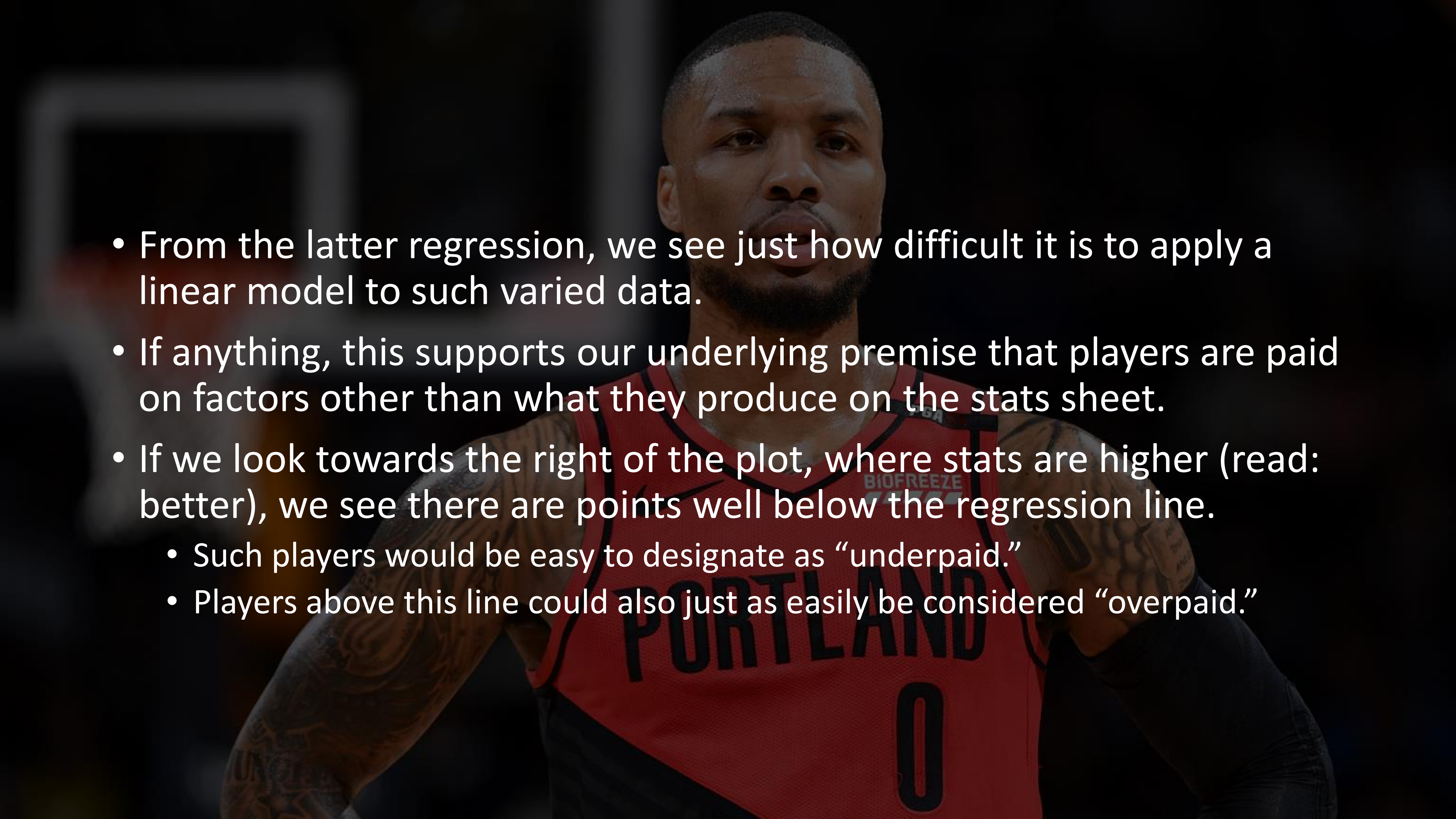
OLS Regression Results

Feature	Coefficient	Std. Error	T	p-value
Intercept	-1.445e08	4.98e07	-2.903	0.004
Year	6.711e04	2.44e04	2.747	0.006
Age	2.153e05	3.53e04	6.094	0.000
Usage Rate (USG%)	5.162e05	7.96e04	6.488	0.000
Win-share (WS)	1.020e06	6.56e04	15.535	0.000
Total Rebounds (TRB)	9.015e04	2.48e04	3.629	0.000
Points (PTS)	-2.167e05	7.45e04	-2.910	0.004

- Our PTS feature had a negative coefficient and probably some collinearity with WS
- Other “value” metrics (advanced statistics such as VORP, BPM, and PER) were put into the regression but did not help the model, only showing up with negative coefficients

Original Data





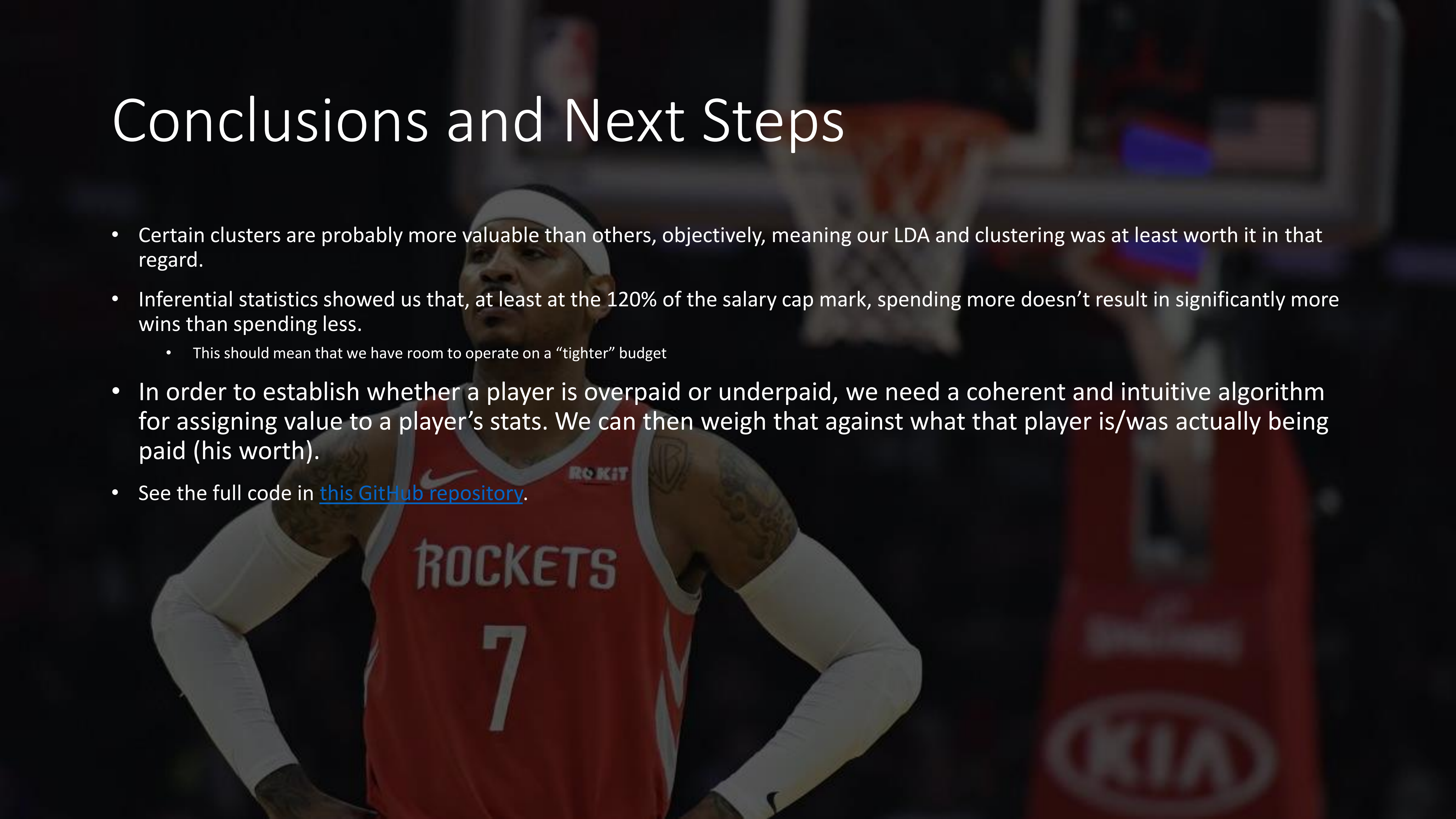
- From the latter regression, we see just how difficult it is to apply a linear model to such varied data.
- If anything, this supports our underlying premise that players are paid on factors other than what they produce on the stats sheet.
- If we look towards the right of the plot, where stats are higher (read: better), we see there are points well below the regression line.
  - Such players would be easy to designate as “underpaid.”
  - Players above this line could also just as easily be considered “overpaid.”

Final Insights: Value vs. Worth

- Where are left is with a philosophical question of "Value" versus "Worth," where the former can be defined as "the dollar amount a player contributes to his organization," and the latter as "what someone is willing to pay for that player's services."

- Our regression model does a sufficient job of predicting player worth (though as we saw, there is still plenty of variance). If we can form a coherent method for assigning a dollar amount to a player's contributions (i.e. his "value") then we have a means of establishing if the player is over or underpaid.

- If the player's value is greater than his worth, he is underpaid.
- If the player's worth is greater than what he brings, his value, he is overpaid.



Conclusions and Next Steps

- Certain clusters are probably more valuable than others, objectively, meaning our LDA and clustering was at least worth it in that regard.
- Inferential statistics showed us that, at least at the 120% of the salary cap mark, spending more doesn't result in significantly more wins than spending less.
  - This should mean that we have room to operate on a "tighter" budget
- In order to establish whether a player is overpaid or underpaid, we need a coherent and intuitive algorithm for assigning value to a player's stats. We can then weigh that against what that player is/was actually being paid (his worth).
- See the full code in this GitHub repository.