# Wide Receivers Performance in the NFL

Capstone Project 1 Final Report

*Conducted under the supervision of Lucas Allen (mentor at Springboard)*

# Our Goal

The goal for this project is to help NFL's general managers to optimize decisions in the recruiting of wide receivers (WR) during the NFL draft. In particular, we provide a deep analysis of which players are best to recruit to satisfy the needs of the team based on their collegiate experience and combine numbers. The data, when complete, would include almost all statistics known of a player prior to their draft and the model would predict one's ability to perform in the professional league.
But what do we mean by **performance**? Originally, our interest was focused on a player's longevity, potentially the total number of receiving yards or touchdowns and if such a metric would meet a threshold set by the team's manager. Finding that our data would be limited, either because older players' performances are difficult to collect or because younger ones have not yet expressed their full potential, we chose our metric as follow: **the total number of yards during the first 5 seasons.**
In fact, given that our data is limited to players drafted between 2000 and 2011, we have complete information during their first 5 years in the NFL. Additionally, a general recruiter is more interested in a player's performances during his first contract, which frequently lasts this amount of time.

### Project summary:

This project required extensive data collecting, as we weren't able to find a resource which explicitly contained each draftee's performance as a college player, during the combine and as a professional player. Nevertheless, we were able to use the following resources.

Pro-football-reference (and its affiliate sports-reference.com): A website which contains all imaginable sets of stats for each nfl players and which is easy to scrap. However, the website presented major difficulties in that:

- frequently, a set of data per player was missing,
- the data was spread among multiple pages,
- the data's format lacked consistency from one player to another.

Kaggle: A resource which contains the combine numbers of all draftees from 2000 to 2017, easily exportable in a csv file, but which unfortunately is filled for missing data, either by default or by error.

Finally, the project and this project report is divided into 3 main sections:

1. **the data wrangling**,
2. **the data story** which includes the inferential statistics report,
3. and **the in-depth analysis:** the formulation of a machine learning based model which aims at predicting our target value depending on the collegiate and combine performances.

*Note to reader:* When we started this project, we had very confidence that we would be able to capitalize on the data gathered and create a model that would predict how well a drafted player would perform in the NFL. As we've advanced in the completion of our project, it came clear that a lot of random events needed to be considered for our choice of target and that the feature data we collected wouldn't suffice in building a robust predictive model. As of this writing, we're still exploring options that would correct such imperfection. Hence the presence of an additional section entitled **future directions**.

# Data Wrangling

Our methodology to obtain a clean data set is simple on principle:

1. Use BeautifulSoup to scrape pro-football-reference and to collect the NFL and CFB data for each player drafted between the year 2000 and 2011. Report the data in a csv file, where the rows correspond to the players and the columns are the nfl and college stats.
2. Download the combine file from Kaggle and save the data of WR from 2000 to 2011 only.
3. Merge the two by draft's year and pick or by name.
4. Clean the data.

## 1. The Draft table

This is the more complicated step. We use the website pro-football-reference to capture the nfl and cfb data in 3 steps. Most of this involves using the package *Beautiful Soup*. Since the tables of draftees and the players' profiles are spanned over multiple web pages, the following work is done through automated functions.

### A. Generate the table of draftees 'draft table':

This table is preloaded with basic information (player's name, draft year position, college) and basic NFL stats (aggregate nfl touchdowns, aggregate yards). It is properly formatted and there's little work to do besides providing the url to the Beautiful Soup package, which scrapes the page and turns it into a Soup document. The difficulty comes down to locating and scraping the table in Soup format using the 'find_all' command, and then turning it into a dataframe. The columns represent basic sports information, but very importantly, also contain two urls:

- cfb url, which redirects to the player's college football profile
- nfl url, which redirects to the player's nfl complete profile.

Note that a moderate proportion of players have the nfl and cfb url missing. For the former, nothing we can do, but for the latter, we generate the url link by concatenating the player's first and last name as follow:
cfb url = 'https://www.sports-reference.com/cfb/players/'+*firstname*+'-'+*lastname*+'-1.html'

We then expand the draft table to accomodate for additional columns, and using a simple for loop, we use the two urls to scrape for additional data. The idea is then to merge this new data in those additional columns.

## B. The CFB entries

We now use the sports-reference website. Using the provided cfb url, we reproduce the same approach than for 'draft table'. In turn, we're able to generate the 'cfb table', a table of 'Receiving and Rushing' football stats at the college level.
We only worry about the overall cfb stats of a player and use an additional function to generate and merge that data into the 'draft table'.

### Issues

- Precaution is needed for players having played in multiple schools.
- There is also a substantial number of players drafted as wide receivers, but who had a different position in college. For those, the 'Receiving and Rushing' stats are a bit further into the web page. Unfortunately, BeautifulSoup is limited in that it cannot scrape all the tables of a web page; in Soup terms, if the table is between consecutive <a! symbols, it is hidden to the package. Therefore, for players whose table of interest isn't rendered by Beautiful Soup, we use the webdriver.Chrome package, which allows us to capture the proper table by name recognition.
- Finally, we must make sure that the 'Receiving and Rushing' table displays the data in proper order: for those with better rushing than receiving stats, we apply an additional function to generate the row of numbers in proper format

Of course, if the program fails to open the page, the function returns NULL entries.

## C. The NFL entries

We reproduce the same approach as we did with 'cfb table' to generate 'nfl table', a table of receiving stats in the NFL. We once again focus on the aggregate numbers and merge the data into the corresponding row of the 'draft table'.
Scraping this advanced nfl data is more complicated than its cfb counterpart, because of the difference in nature of each player's nfl career. A majority end up playing as WR, but some do not play at all, some get to play very little and some are drafted as WR and yet play at a different position in their career. This abundance of cases complicated the scraping process and required multiple sets of data collecting / cleaning summarized in the following functions:

1) main0: making no assumptions whatsoever, generates the 'nfl table'
2) main1: verifies the title of the table match with 'Receiving and Rushing' stats; otherwise, it uses the webdriver.Chrome package to search for other tables
3) main2: verifies the receiving yards generated are correct
4) main3: manually investigate the data generated and search for errors

### Other Issues

- The aggregate row isn't necessarily the last row of nfl table (e.g. players having played for multiple NFL teams).

- We make sure that the receiving and rushing stats are properly ordered.

Of course, if the url link provided fails to open a page, the function returns NULL entries.

## 2. The Combine table

This step is much easier. A csv document downloaded from the Kaggle website contained the combine data for all the players drafted between 2000 and 2017. We only need the sets of WR from the 2000 combine to 2011, included. To do so, we used a simple boolean conditional on the column 'Year' and the column 'Pos'.
Later through our project, we realized that some of the players drafted as WR didn't enter the combine as WR. Therefore, we remove the position conditional of our boolean method.
The result is a data frame of combine numbers for wide receivers. We export the file as 'combine table'.

## 3. Put it all Together

### Merging

Once we've generated the 'combine table', we add additional columns to the 'draft table' corresponding to the combine data, and match the two sets of data by year and name or by year and draft's position.

### Searching

Since there is a substantial amount of players with no combine data, we use the nfl url from 'draft table' and generate missing combine data from the Pro-football-reference website using the webdriver.Chrome package. We then carefully reformat the entries to match those generated by the merging method mentioned above.

### The Method Columns

Through these data collecting steps, for each player and for each set of stats (nfl, cfb or combine), we also note the generating method used every single, which we store in additional columns of our extended data frame. These notes serve as comments for the user, to know how the data was generated and how complete it is.

#### The 'cfb method' column:

- success: cfb data is good
- fail: cfb data is missing

#### The 'nfl method' column:

- success from main, main1, main2, or main3: nfl data is good
- fail (eg link broken): no nfl data

- fail after main1: player never had 'Receiving & Rushing' stats (he was used for something else)
- fail from main2: player with mismatch between ‘'receiving yards' and 'nfl yards' . There shouldn’t be any of those

The 'combine method':

- success from merge, search: combine data is good
- fail after search (missing table): no combine table was found on the nfl web page
- fail after search (missing link): no url was even provided

### Summary

| combine method<br>cfb method | fail after search (missing link) | fail after search (missing table) | success from merge |
|---|---|---|---|
| fail (eg link broken) | 8 | 14 | 31 |
| success | 10 | 30 | 300 |

Out of 393 data points, we have a total of 300 with complete cfb and combine data, while 31 have combine data only, 40 have cfb data only and 22 have neither.

### Generating Additional Data

Since the basic nfl data only contained aggregate career numbers for each player, we figured it would be necessary to put to good use the complete nfl statistics we generated for each of them. In fact, if we’re interested in performances during the first years of a player’s NFL career, we would need to get the year by year data of each player. We do so using the ‘nfl table’ mentioned above.
We can now generate two new metrics:

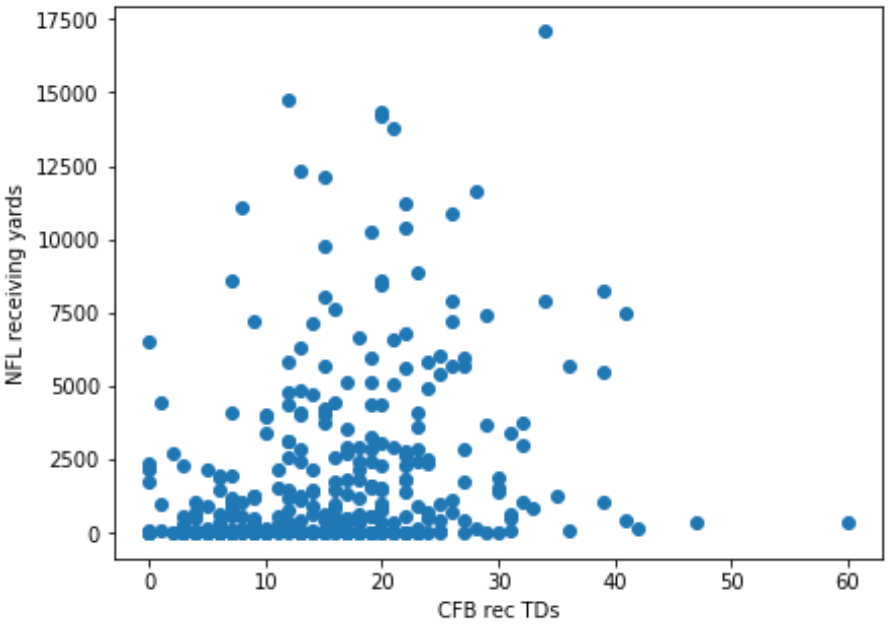
- # of seasons with yards > threshold
- # of rec yards during the first 5 seasons

## 4. Data Cleaning

Most of the data cleaning consisted of the following steps:
- Re-organizing the csv file to be more understable: renaming columns, changing order of columns.
- Replacing error messages with np.nan entries for missing data and with the value ‘0’ when we’re certain of it (when a player doesn't have any rushing stats for example).
- Catch leftover errors in the nfl and cfb data via automated functions or manually
- Carefully convert all numerical values to floats with a few exception (such as years and draft positions)

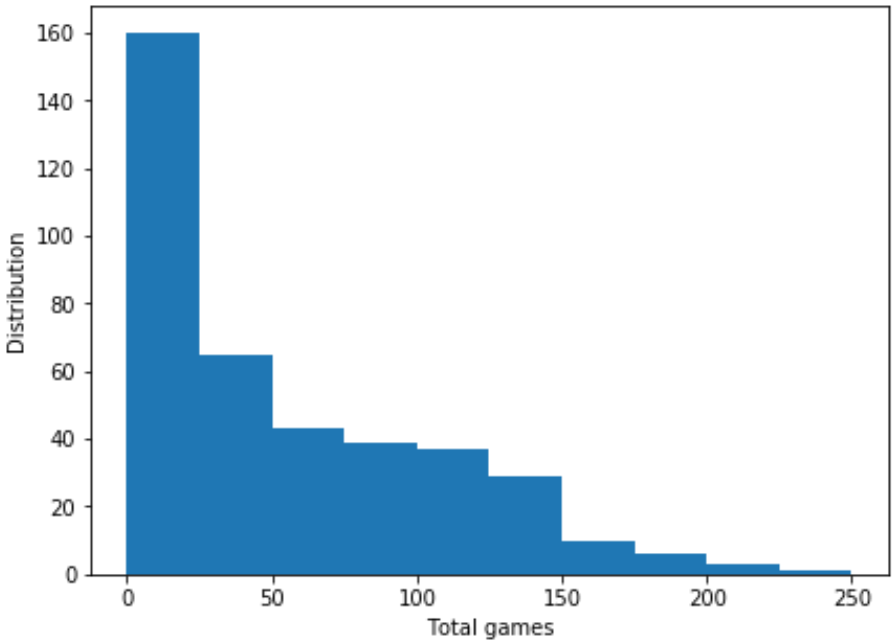
# Data Story - Statistical Analysis

Now that our data frame is cleaned and saved in csv format as 'wr_tdc_new.csv' ('wr_tdc_clean.csv' missed the last 2 metrics we generated), we can finally explore some of the trends in players' performance and derive rigorous statistical tests aimed at highlighting the impact of categories on our target metric.
In fact, we're now looking to put in evidence how the cfb and combine data impact the nfl data. Of course, our ultimate goal was to narrow it to a specific target, but to start with, we explored the more evident numbers: the number of touchdowns and the number of receiving yards.




On this scatter plot, we discern that the nfl receiving yards increase with either the number of receiving touchdowns in college, or the number of receiving yards. But what's striking even more is the concentration of data near the x-axis, suggesting that even some of the most performing athletes in college can get little to zero scores in the NFL.

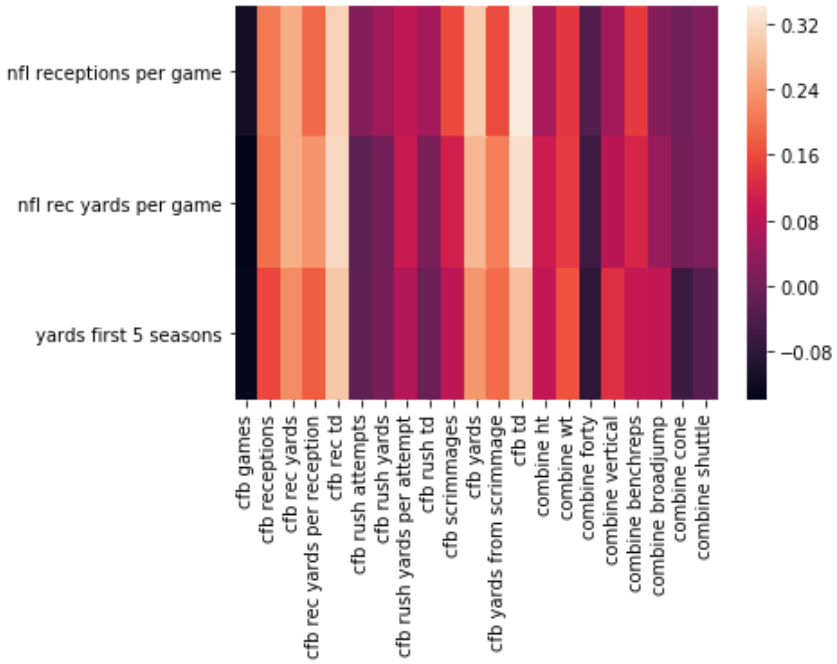
This has to do with the amount of playtime given. In fact, the histogram of number of NFL games confirms that a lot of players simply don't ever get to play much.



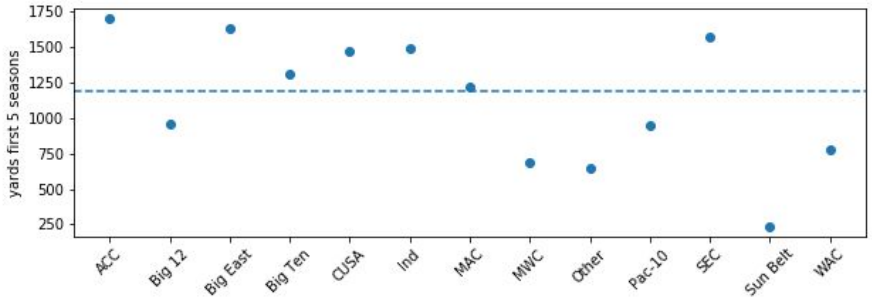
## Correlation Matrix and Non-numerical Data

Given the extensive amount of variables we have (a total of 37 nfl stats and 24 cfb/combine stats), we decided to compute the correlation factor between each of our potential targets ( $Y_i$ ) and the input variables ( $X_j$ ): $C_{ij}$ . Of course, this can only be done for numerical variables. We then derive the aggregate correlation factor for each nfl variable, and select those with the highest value - those that correlate the most with our cfb and combine data. We pick 3 nfl variables as our potential targets and plot the correlation of those with the rest of the data:

- NFL receptions per game
- NFL receiving yards per game
- Number of yards during first 5 NFL seasons




We also consider our non numerical variables: the conference of origin (represented in the figure below) and the class status of a player upon entry in the draft. We only show the results for the number of yards during the first 5 seasons (for reasons explained below).



All three potential targets present very similar results. As we've stated at the beginning of this report, for equity purposes, it is best then to choose the target $y$ ='yards first 5 seasons'. The choice of an overall career metric instead would be falsified, since we're dealing with players drafted up until 2011 and who are at the midpoint of their NFL career.

## Statistical Tests

We run three statistical tests to emphasize the correlation (or the lack thereof) between our target and a set of input variables. We pick a level of confidence of $\alpha = 95\%$ for all tests.

1. Test1: College Football Conference / SEC vs Big12

We run a bootstrap inference on those two categories and we do it with respect to the mean of each of our metric $y$ . The null hypothesis can be stated as "The sec and big12 samples are from the same distribution with respect to the mean of $y$" and found the p-value: p = 0.0213

2. Test2: College Football Rush Attempts

Group1 and group2 are the subgroups of players with cfb rush attempts respectively below and above the median. This time, we test for the independence of the two samples to our target using a scipytest and find: p = 0.3569

3. Test3: cfb receiving touchdown

Group1 and group2 are the subgroups of players with cfb receiving touchdowns respectively below and above the mean. We follow the same procedure as test2 and find p = 0.00000060

Test1 and test3 return on average relatively small p-values which indicates that under the stated null hypothesis, the probability of returning a result as extreme is very small, meaning we can dismiss the hypothesis. This confirms what we saw on the correlation heatmap: that some variables have l and the plot. Test2 on the contrary returns large p-values, meaning the null hypothesis is likely to be correct, once again confirming what we saw on the correlation heatmap.

# In-Depth Analysis

Despite our effort to collect and organize our data set, we must acknowledge that it is pretty challenging and most importantly... limited. For recall, our target is **y = yards first 5 seasons** and we have a total of **23 input variables**, including two non-numerical features: the conference and the class status upon entry in the draft. Unfortunately, we only have a total of 393 data points and only roughly 75% of the numerical data is complete - 300/ 393 to be exact. If you account for the discrete features, this number even drops to 273/393...

But most importantly, very noticeably, our data is very spread out. A lot of players end up having 0 play time in the NFL, either because of lack of competitiveness or because of injuries, despite very satisfying performances in college football or in the combine. For example, as we've indicated earlier in this report, $X =$ cfb_td (the variable which correlates the most with y) has an impact on most of player's NFL performances, but for a substantial number of them, it doesn't $y = 0$. This fact is precisely what makes building this predictable model particularly hard. Hence, most of our strategy consisted in **applying transformation to our data set** (by either reducing or augmenting the number of features or rows) and by using **different linear regressors**.

## Our metric for model tuning

As we attempted multiple supervised machine learning models on the versions of our data sets, for each attempt, we computed the coefficient of determination $R^2$ and very importantly, the root-mean-square deviation **RMSE** which we looked to minimize. We derived those two metrics using a cross validation test with 10 folds and returned the average values for each of those. As we went on with our attempts, it is worth notice that we also had to consider the average **CV score** averaged over 5 folds, for reasons we explain in the next section

## Multiple attempts to improve Algorithm's Performance

It is worth noting before we started all our tryouts a total of 5 feature columns from the combine results exhibited more missing data that their counterparts:
'combine_vertical','combine_benchreps','combine_broadjump','combine_cone','combine_shuttle'

Given that those have very little importance in a player's overall combine performance and that a lot of them do not attempt those combine tests, we simply got rid of these variables to reduce the dimensionality of our feature variable **X** to 18 columns.

We then proceed to apply the **LinearRegression** regressor from Scikit-Learn on different versions of the data:

### 1) All the data

By considering X entirely and dropping the missing entries, we would return a root-mean-square error averaged over 10 folds of RMSE = 1515.6908. Given that number of yards range between -1.0 to 6201.0, this was a substantial number

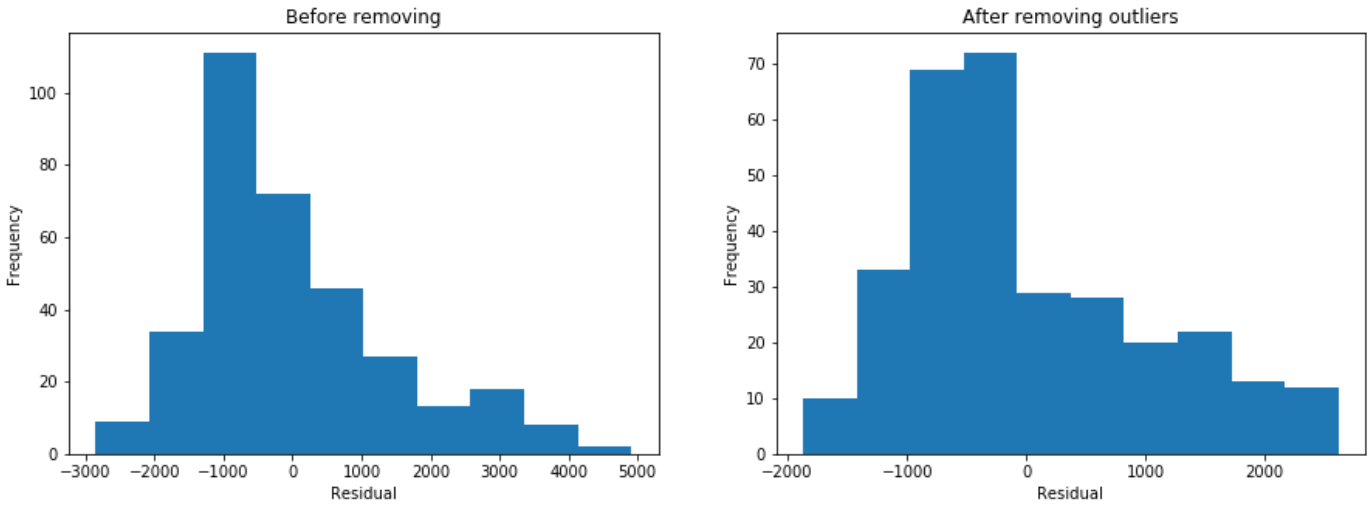
### 2) Reducing X to 6 essential features

We then reduced the features variables to the 6 most essential, those that correlate the most with our target y, namely:
**X'** = ['cfb_rectd', 'cfb_td', 'cfb_yards', 'cfb_recyards', 'cfb_yardsfromscrimmage', 'cfb_games']

we were able to reduce the root mean square error to $RMSE = 1437.9412$.

### 3) Removing outliers

We then decided to use the **ordinary least-squares** function of Statsmodel in order to remove some of the outrageous outliers of our model. We picked those outliers to be defined by having a residual greater than 2,500 when running a simple linear regressor on the entire data (rather than by a train/test split method). We additionally plotted the distribution of these residuals before and after removing those data points:




Yes, the residual distribution wasn't nearly as spread, but most importantly, they weren't normally distributed, indicating once again how challenging the data set is. Still, we were able to reduce that number the RMSE to: RMSE = 1363.6585 when running our CV test on a train/test split data.

### 4) Non-numerical Features

We then considered the features X' along with the non-numerical variables:

Cfb_conference, cfb_class

For the sake of the linear regression, we were able to turn those features into numerical variables using the **get_dummies** function of Pandas. But unfortunately, we did not observe a reduction in the residual.

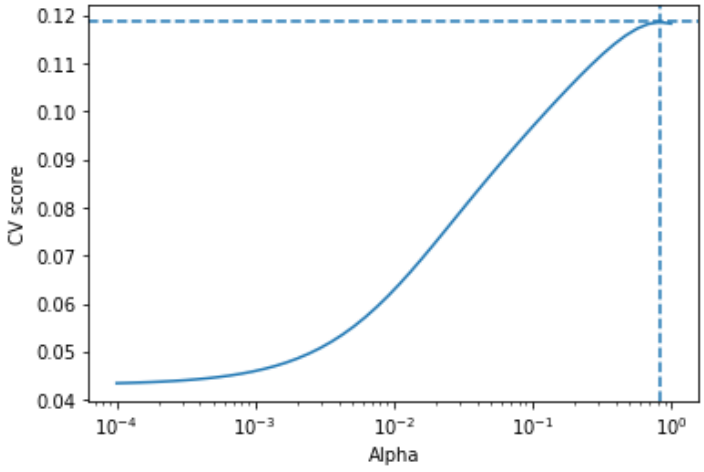
### 5) Imputer

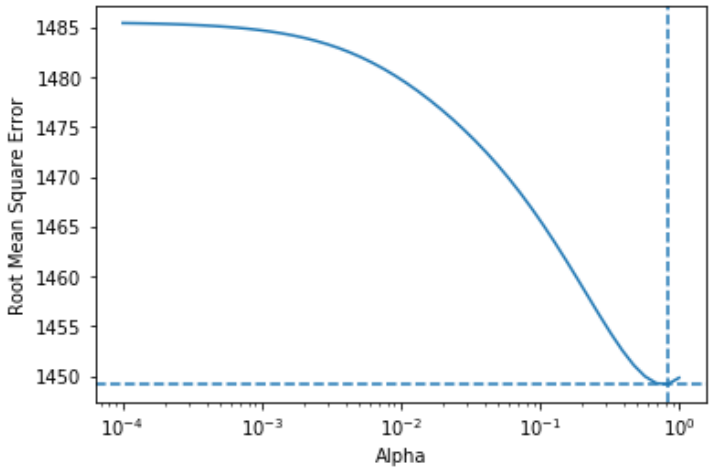
Finally, in order to handle the missing data, rather than removing each row which included a missing number, we applied an imputer from **SimpleImputer** (Scikit-Learn) which would fill our feature data using a *mean value* based strategy. Doing so, we were able to find a modest RMSE = 1383.1872.

But using variations of our data set wasn't our only objective; we also used the **Ridge Linear Regressor** from Scikit-Learn to build our regression model. The strategy was to be able to minimize the residual or to improve the cv score by tuning the $\alpha$ parameter of the regressor. We did so in two different circumstances:

### 6) Ridge Regressor on all data

With our original data and without applying the imputer, we found the RMSE to reach its minimum at 1449.1586 for $\alpha$ = 0.8286. For this same parameter, the cv score returned was then = 0.118580




### 7) Normalizing and Nonlinear data

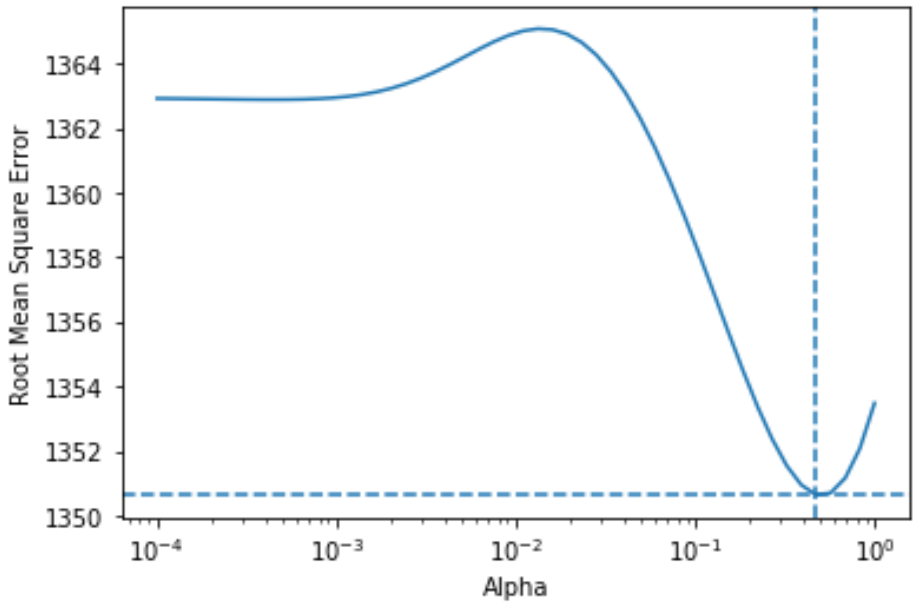
Given that some input statistics were not on the same scale (for example, cfb_td would range between 1 and 61, while cfb_recyards would between 2.0 and 4414.0), we decided to normalize all feature variable X. Unfortunately, the best cv score was identical than in the 6th attempt.

We also assumed, that given that the number of TD's in college was the most important feature variable, we could suspect a nonlinear behavior between such and the target y. Which is we decided to run the linear regressor on the square of cfb_td but found once again that the best cv score to remain stagnant: score = 0.11195 (note here, the best parameter for $\alpha$ was then = 0.68664)

## Final Model

By considering all these attempts, we came up with our final algorithm and final data set, which included:
- choosing the most important features X' along with the non-numerical variables (conference and class)
- remove the outliers
- applying an imputer with mean value based strategy to handle the missing data
- run a ridge linear algorithm

The returned RMSE was 1350.646




## Future Directions

Throughout this project, we hoped to design a predictive model based on the collegiate and combine performances, which, when fed a particular data set, would return stasifying prognostics on the number of yards by a drafted WR during its first 5 years in the NFL. Unfortunately, we weren't able to come up with satisfactory results.