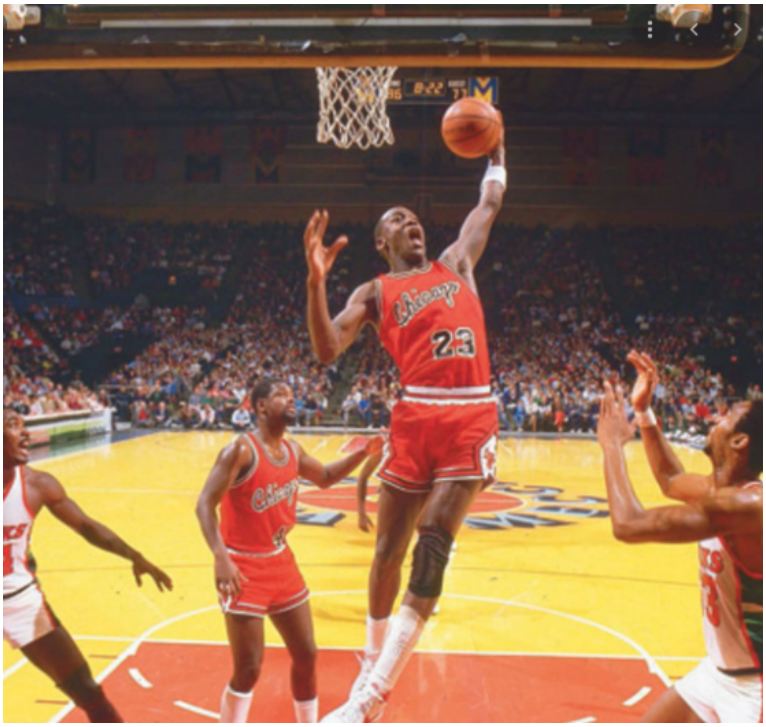
Predicting NBA Center Rebounds Over/Under Betting Spreads



Mario Hage

I - Introduction

With the rise of gambling, and specifically online gambling in the past couple of years comes an interest to attempt to gain an edge over certain situations. Naturally, because the gambling in this project will revolve around sports, I chose to drill down into specifically NBA Center Rebounds to try and gain an edge in a specific realm of a medium. The reason for specifying Rebounds is the smaller chance of a player having an off-night, as compared to example betting on points or other stats. The point of the project is to consistently try to out-bet the variance involved in the nature of this type of gambling, and my goal is to consistently return 10-15% on bets using previous NBA Center player data as well as calculating in depth and out of the box features to gain an

edge. The end user intended for the project is for someone who is code savvy and also has a knack and general understanding for basketball/the NBA in general and can look at the results from the project and tailor the information to their needs given different gambling options/spreads/scenarios.I am not looking for this project to back a plan I have to gain that edge over the variance involved in betting; rather, I am looking for this project to provide me the skeleton for that plan. In example: The models may not be accurate at predicting true positive scenarios, however are accurate at predicting true negative scenarios. Since we are betting, we can use the latter to our advantage simply by betting the corresponding bets to a true negative prediction scenario. IE: Betting an NBA center will grab 10 or less Rebounds rather than 10 or above (<= 10 scenario is the false outcome, what matters is we predict a true outcome; the outcome itself does not matter whether is a true negative or positive)

II - Data Wrangling

The nba_api https://github.com/swar/nba_api contains many different endpoints that are extremely useful and well thought out, and I used that API to access all player box score histories (every row contains specific stats pertaining to the respective game that was played.) The specific endpoint that was used was the playgamelog. More information on that can be found here:
https://github.com/swar/nba_api/blob/master/docs/nba_api/stats/endpoints/playergamelog.md
However, because I specified that I would only be including NBA center positions in the project/model, I found a dataset containing a list of all current active NBA Players' names as well as their respective positions. I downloaded and filtered for any Center positions. From there, I used those full names in a loop with the nba_api call to obtain only Center position data. I also felt inclined to add distance traveled as a feature.To get distances travelled between games, I found an external dataset which contained relative time travel durations from NBA cities, and all the possible combinations of travel

in a matrix https://www.rostrum.blog/2018/12/24/nba-travel/ . I uploaded that into my notebook, and I realized that I needed to join the starting and ending cities with my current dataset. All I had as a starting point in my original dataset was the match up column IE: (LAC @ BKN) (Los Angeles Clippers at Brooklyn nets) I sliced that column into Team, and Opponent. From there, I also created another two columns (Previous Status of Last game (Home/Away) and the Previous Opponent using the (shift) function in pandas.This had to be done in order to correctly calculate distances, as the challenge was figuring out the logic for what distance traveled should be in the 4 different scenarios that could occur with Home/Away games and Previous Home/Away Games.

The dataset that was downloaded looked like this: (901 rows)

Start	End	Duration (	Duration (hrs)
MIA	POR	3459	58
POR	MIA	3457	58
GSW	BOS	3311	55
BOS	GSW	3307	55
POR	BOS	3302	55
BOS	POR	3296	55
SAC	MIA	3293	55
MIA	SAC	3286	55
GSW	MIA	3286	55

It contained every possible start and end combination between all the nba team cities.

For example, if a specific team's last 2 games were home games, then they traveled 0 distance, or if their last 2 games were away, their starting destinations would have been the prior away team city. Once I had the logic built out, I used the (*if and)* statements to correctly classify and create new start and end destination columns to be able to merge it with the dataset containing start and end NBA cities and the respective distances between them that I had downloaded previously. I performed a left join on the start and end columns to get all the data in place.

In terms of data history,the amount of games per player collected varies from player to player, as the dataset consists of all games played by current active centers. Due to the nature of what I was provided, just previous game stats cannot be used as features, so I created rolling average columns for minutes played in the last games, to see if fatigue had an effect on Rebounds in a player's upcoming game. I also calculated rolling average Rebounds in the last 5 games to see if a player's "form" or "heat level" has an effect on their Rebounds grabbed in their next game as well. Sticking to "form" or "heat level" I created a column with a player's wins in the last 5 games as well in hopes of finding it to be relevant to our target (Rebounds).

Thankfully, if a player did not participate in a game, then they were not included in the dataset originally so little cleaning had to be done to the data other than the last row of our marged table which contained NAN values for the rolling average columns created earlier as (Shfit) was used and there was no data below the last row.

The final features that were included in modeling were: 'WinsLast5Games', 'rollingavgminutesLast2Games', 'location_Away', 'location_Home', 'Duration (hrs)', 'PreviousHome/Away_Home', 'PreviousHome/Away_Away', 'rollingavgREBscoreLast3Games', 'Opponent' (Dummies)

The reason why the Rolling Average Rebound Column is a Score is because I divided the player Rebounds per game by the rolling average Rebounds over 50 games, and then took the rolling mean of that to prevent data leakage, and to also normalize that feature since the dataset contains data on all the different centers in the league.

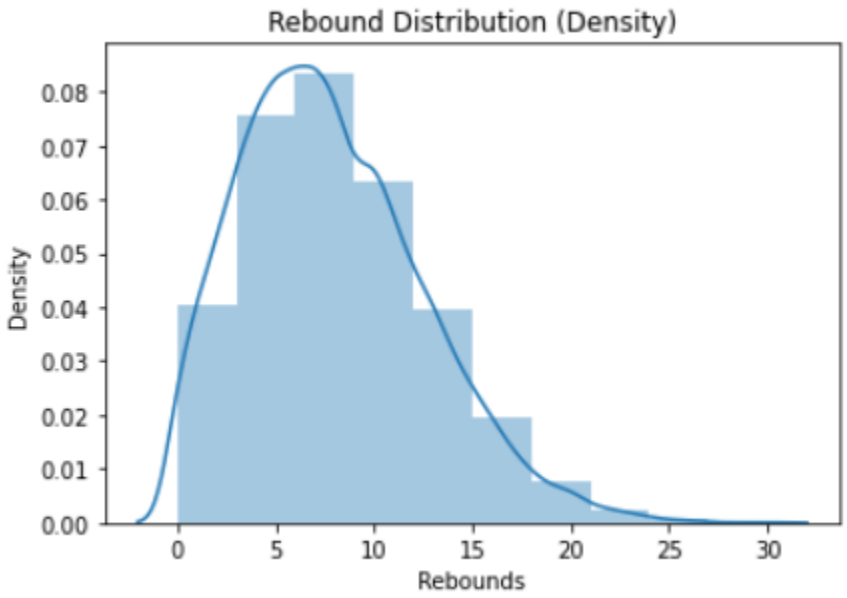
I decided to frame the problem in a way that can be solved using classification, as gambling odds are often over/unders of some sort. In this case, I decided to use 10 Rebounds as the threshold for over unders on Rebound predictions. It is a little higher than the mean for the dataset, but closer to and a little higher than the thresholds for some better websites just to be on the safer side. For example. our threshold is 10 or

less, and some websites may have the odds posted at 9, which would favor whatever our model results with.

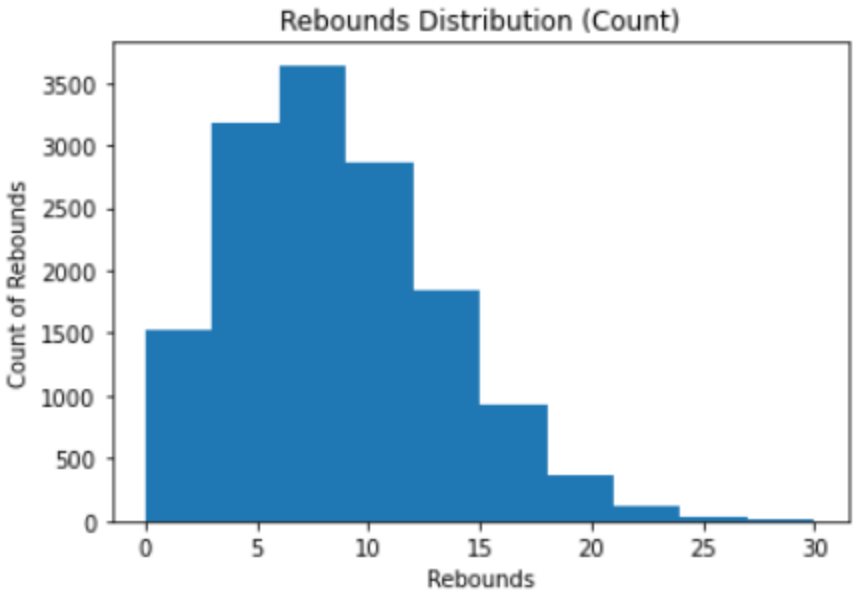
III - Exploratory Data Analysis

Note: Target Feature Analysis is at the end of EDA portion

Beginning with an overall density plot of our target variable (Rebounds): Per the plot below, mean Rebounds for all current active NBA centers is 7.85. Note: Historical data fluctuates depending on each player. It is important to note again that each player's full game history is included in the dataset below. (16,310 games worth of data.)



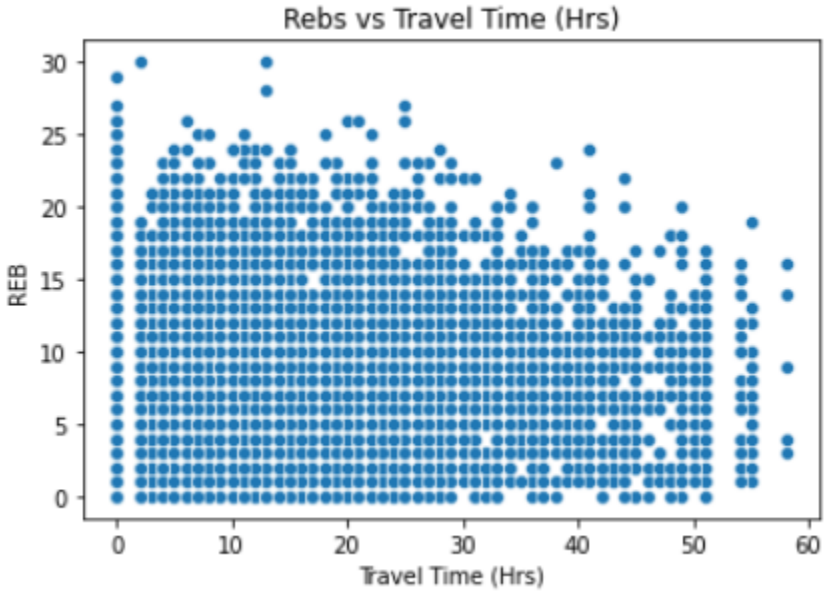
And:



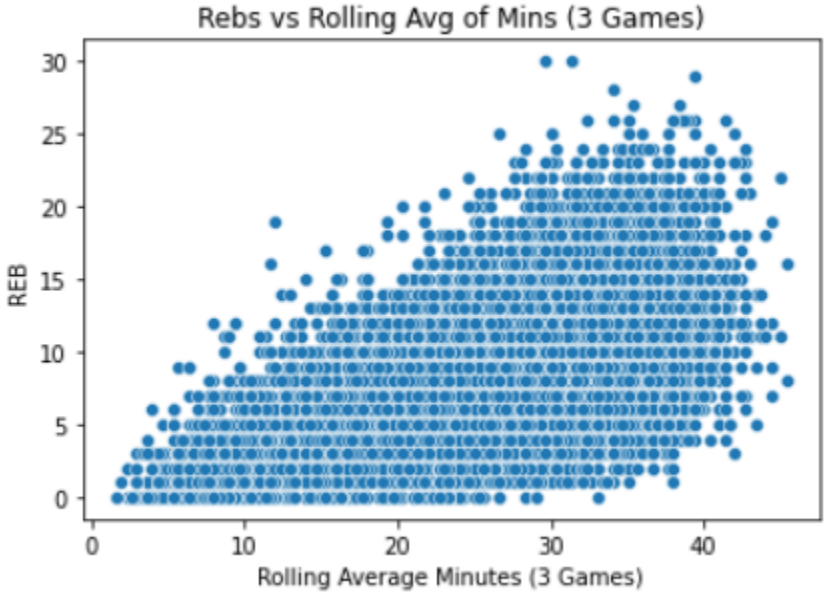
Distribution Counts in the figure above to provide a little more context. The data is skewed to the right, but it does not seem like there are any outliers; except for the rare spectacular performances with 25+ Rebounds. Unfortunately, we do not get a larger pay off if more Rebounds are grabbed. What matters in our case is just passing the 10 rebound threshold. The same thing applies if you are betting Under 10. I guess we can't have our cake and eat it too.

The graph below shows a scatterplot of Rebounds vs. Travel Time. What is interesting in this plot is the clear decreasing maximum values at every 10 hour interval increase.

However, this chart alone cannot give us a clear understanding of the two variable relationships as there are simply less occurrences of game counts as travel time increases.

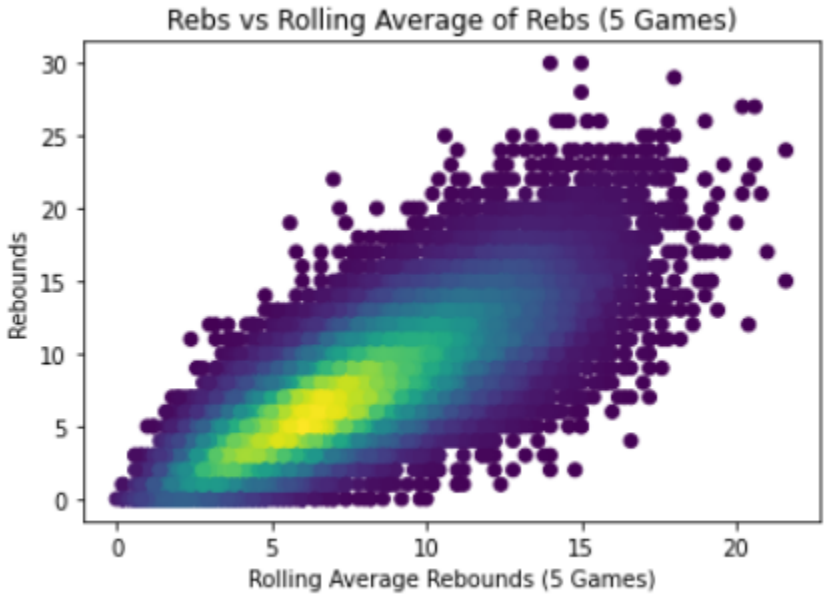


The next chart is a simple scatterplot of Rebounds vs Rolling Average Minutes In the past 3 games. The chart is telling us “Do not bet overs on a player if they have averaged more than 40 minutes in their last 3 games.” As you can see: lower maximum values disregarding one outlier starting at 33 minutes.

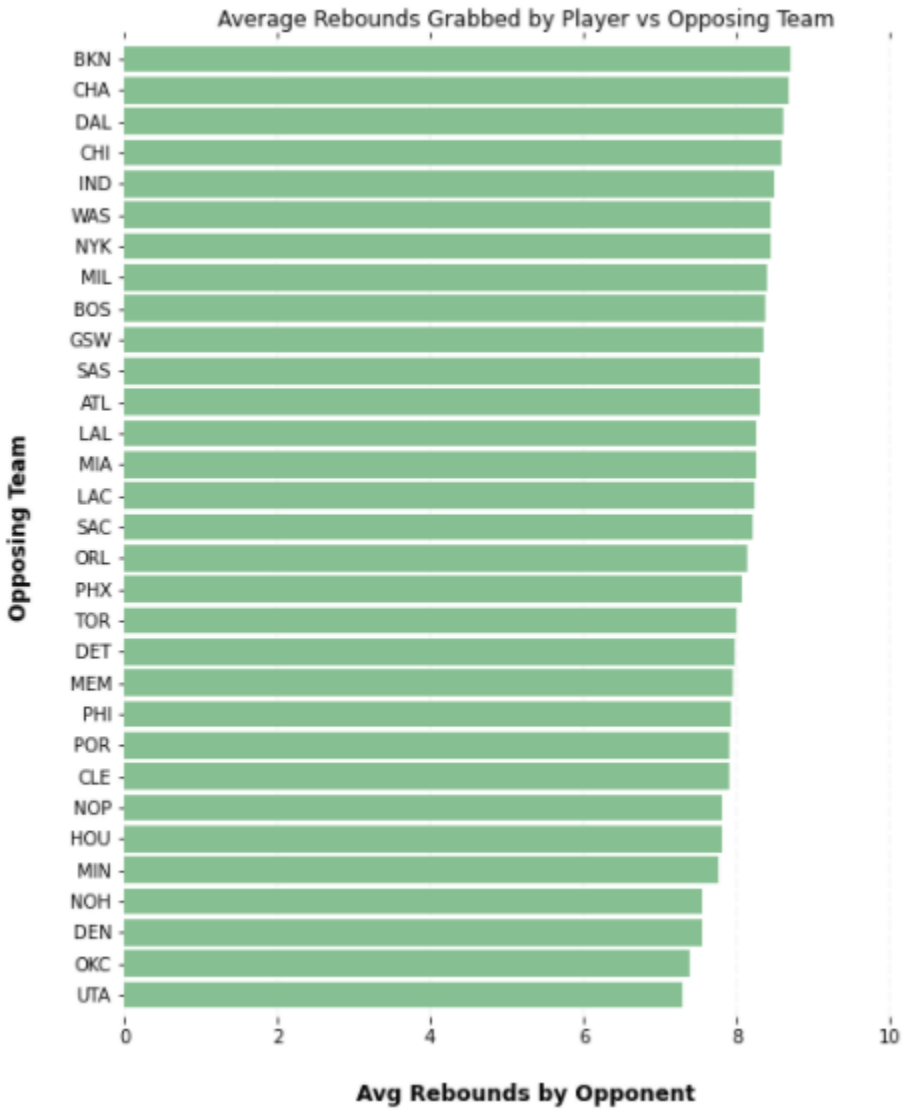


From the charts alone, it seems like at extremities we are able to see clear cut off ranges for Rebounds grabbed, perhaps hinting that we might want to favor betting unders. This is of course dependent on our model’s performance later in the project.

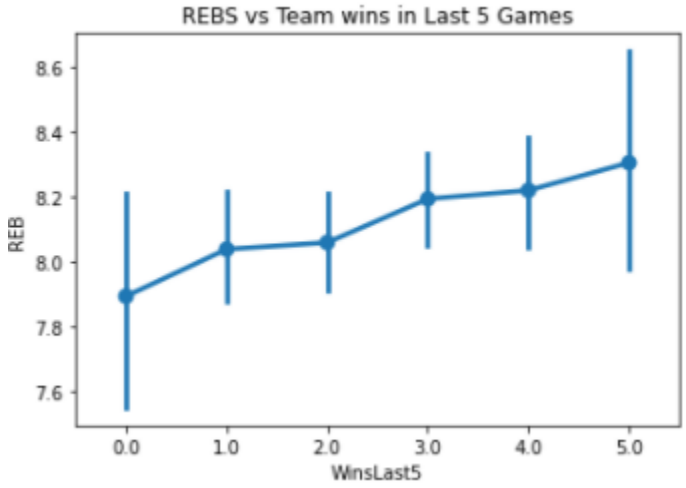
The below chart shows another scatterplot with Rebounds vs Rolling Average Rebounds in the past 5 games for each player, it is also shaded on density; lighter being more dense. What is interesting in this graph is that there are zero instances where a player grabs less than 10 rebounds if they had averaged 17 in the past 5 games. Though it is a quite rare event, if it does happen; it would be smart to trigger a notification signaling an over bet. Of course, one has to beware that this is not 100% fool proof, as injuries or off nights can still occur; but have not occurred in that scenario *yet.*



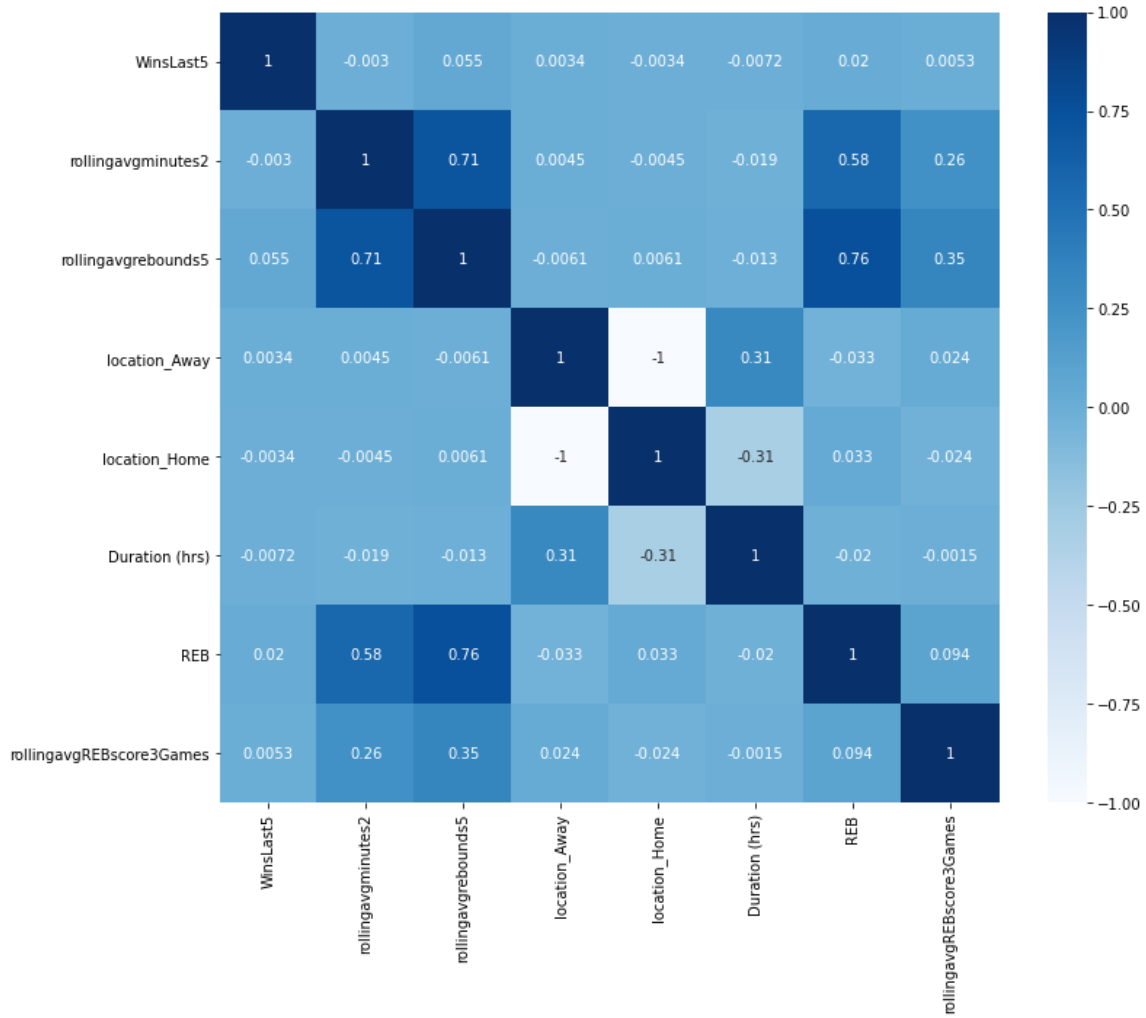
Another interesting discovery was the following bar graph: Average Rebounds Grabbed by Player vs each Opposing Team. It looks like Players across the league struggle to grab rebounds vs UTAH as compared to BROOKLYN; where they grab the most rebounds on avg. The difference is vast, roughly 1 rebound, and in a betting spread where even 1 rebound counts is perfect for a strategy to utilize that information to maximize profit. The difference 1 rebound makes is anywhere between an 8-15% edge over the long term dependent on player sample size mean rebounds.



Below is a pairplot with a team's win's in the last 5 games, and rebounds grabbed by a player. It is interesting, because as wins in the last 5 games increase, so does % chance a given player grabs more rebounds. It is good to note that a team's "form" or measure of how on target a team is is measured by Wins in the Last 5 games here, and it seems to have a small correlation with players grabbing more rebounds. What would be interesting is gaining the funding or putting in the extra time to put more of a detailed score on a team's "form," it would be interesting to see all the creative ways one could come up with that feature.



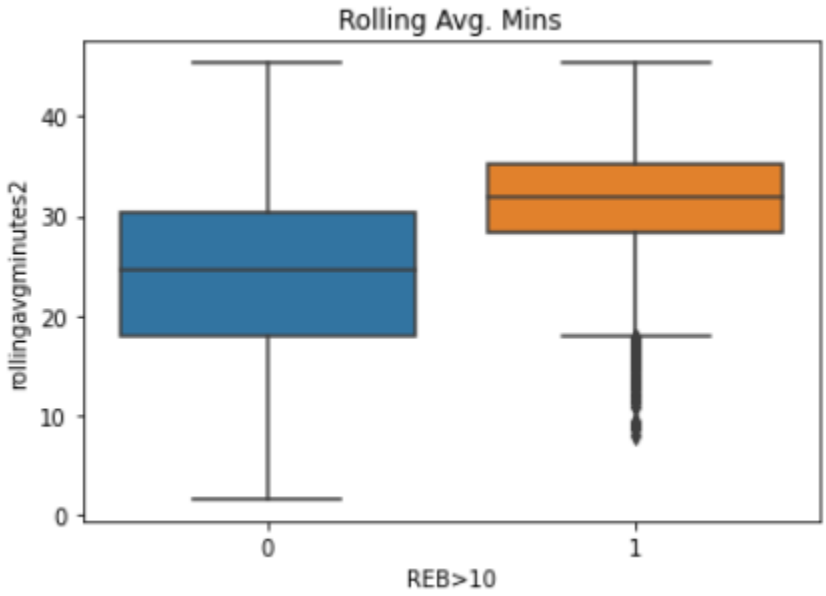
Finally, a heatmap with the correlations between the features chosen for modeling:

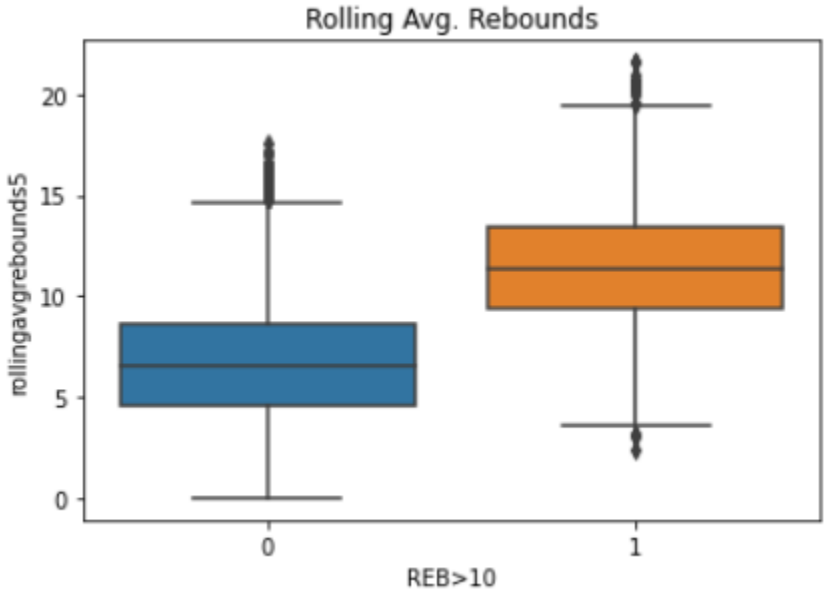


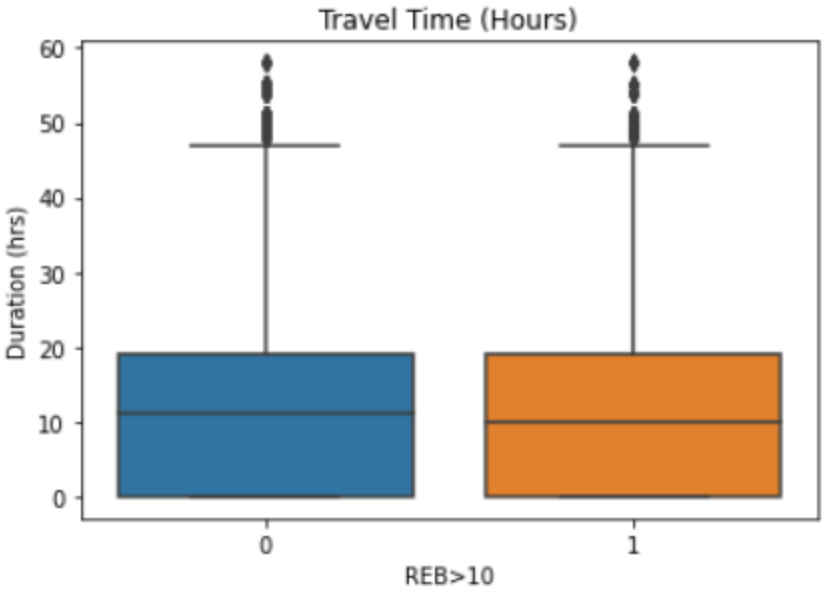
The darker shade blue indicates a more positive correlation, and the colors lighter than teal indicate a decreasing negative correlation as the color gets lighter. From the heatmap, it looks like the rolling average scoring over 3 games has a 0.094 correlation with rebounds, duration has a -0.02 correlation, the rolling average for rebounds in the past 5 games has a 0.76 correlation, rolling average minutes played in the last 2 games has a 0.58 correlation. The 2 most highly correlated features do not tell us much, as it is an expected increase. However, I think if we can frame the problem differently, we can use these numbers to our advantage. Though we could not find the correlation strictly

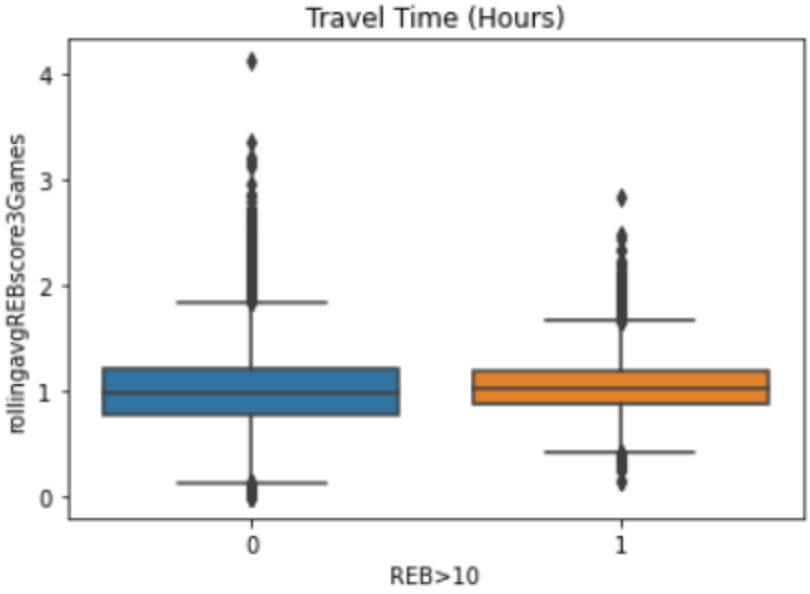
with rebounds, the features may tell us more if we transform the problem. It definitely looks like an above or below rebounds threshold classification is the right way to go with modelling because of this.

EDA - Target Feature Analysis (Viz)









IV - Feature Importance

After plotting the correlation heatmap, I ran a Random Forest Classifier to get the following feature importances. I have decided to use all of the following features. I am not too worried about the below importances, as we just need the model to accurately predict one side of the classification, and it does not matter which; as we can find a way to profit from both sides of the equation.

```
{'WinsLast5': 0.050758547749473554,
 'rollingavgminutes2': 0.17884414268054308,
 'rollingavgrebounds5': 0.3788050152290544,
 'location_Away': 0.00649726837373837,
 'location_Home': 0.006214753874765955,
 'Duration (hrs)': 0.06870002340418552,
 'PreviousHome/Away_Home': 0.009368823598495439,
 'PreviousHome/Away_Away': 0.009395654109252465,
 'rollingavgREBscore3Games': 0.14556581966411472}
```

I decided to also include both rolling average rebounds in the past 5 games as a feature alongside the scored version, as it would benefit to feed both player average data, and score. We know that it may buff predictions, however that is the intent; as odds are also changing player by player.

V - Modeling

A training/test split of 0.9/0.1 was used across all models. The reason being is that the data is consistent across time, and I wanted to minimize the size of the test set to get as close to real time as possible.

i) - Random Forest

When testing the Random Forest Model, I grid searched and cross validated the following parameters to find the optimal parametersfitting the data

```
param_grid = {
    'n_estimators': [100,500],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth': [3,4,5,6,7,8],
    'criterion': ['gini', 'entropy']
}
```

Inputting them in the model and then fitting the data

```
rfc_cv.best_params_
```

```
{'criterion': 'entropy',
 'max_depth': 8,
 'max_features': 'auto',
 'n_estimators': 500}
```

```
# instantiate another RFC with optimal parameters, fit, predict, and calculate recall
rfc_best = RandomForestClassifier(random_state=42, max_features='auto', n_estimators=500, max_depth=8, criterion='entropy')
rfc_best.fit(X_train, y_train)
```

Mean AUC score using cross validation:

```
[0.8739944  0.89542272 0.88359206 0.88048515 0.87327372]
Mean AUC Score:  0.8813536115475198
```

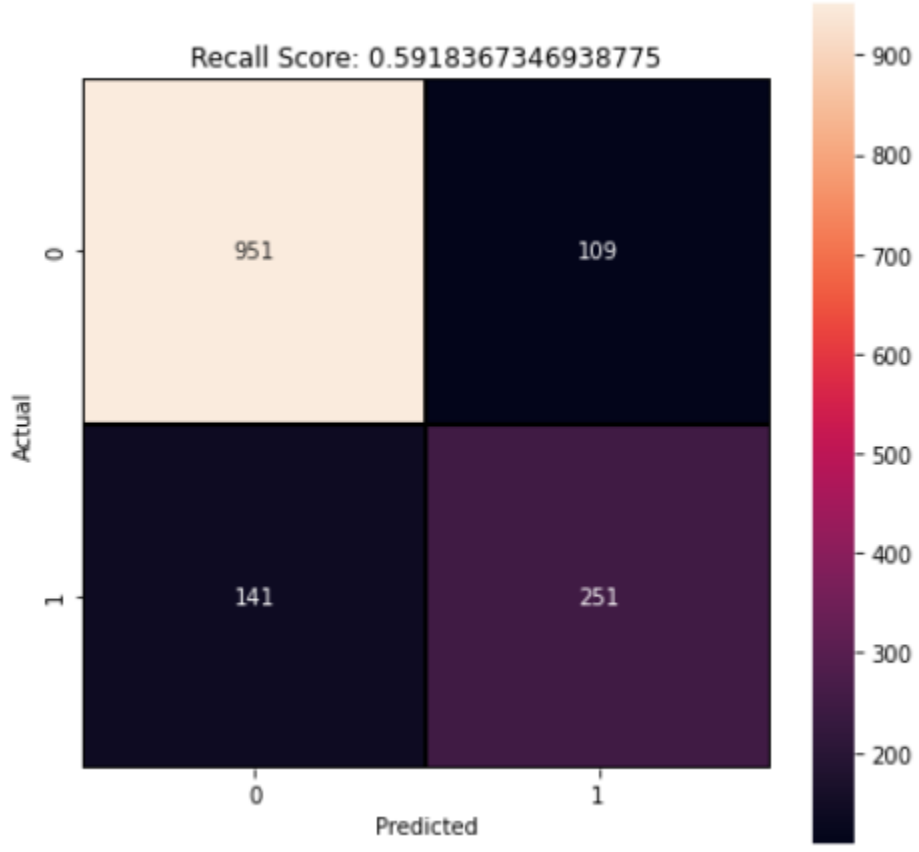
The below classification report shows the precision, recall, and f-1 scores for 0 (Player grabbing 10 or less rebounds) vs 1 classifications (11 or above rebounds grabbed.) It looks like the model is accurate at predicting unders (0.91 recall on 0 classifications.) It is important to note that mean rebounds is 7.85, and our threshold is 10 and to take that

into consideration accordingly when using the models in this project for betting purposes.

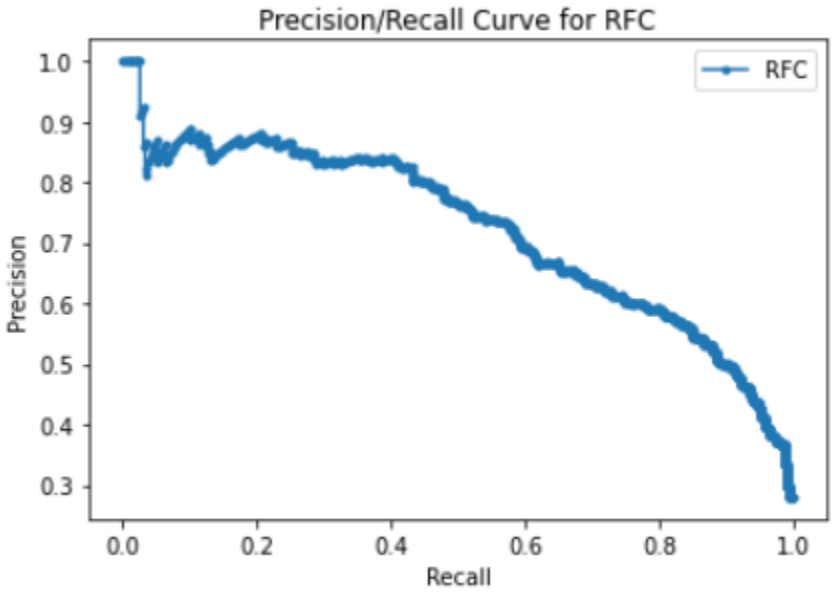
Classification Report

	precision	recall	f1-score	support
0	0.86	0.91	0.88	1060
1	0.70	0.59	0.64	392
accuracy			0.82	1452
macro avg	0.78	0.75	0.76	1452
weighted avg	0.82	0.82	0.82	1452

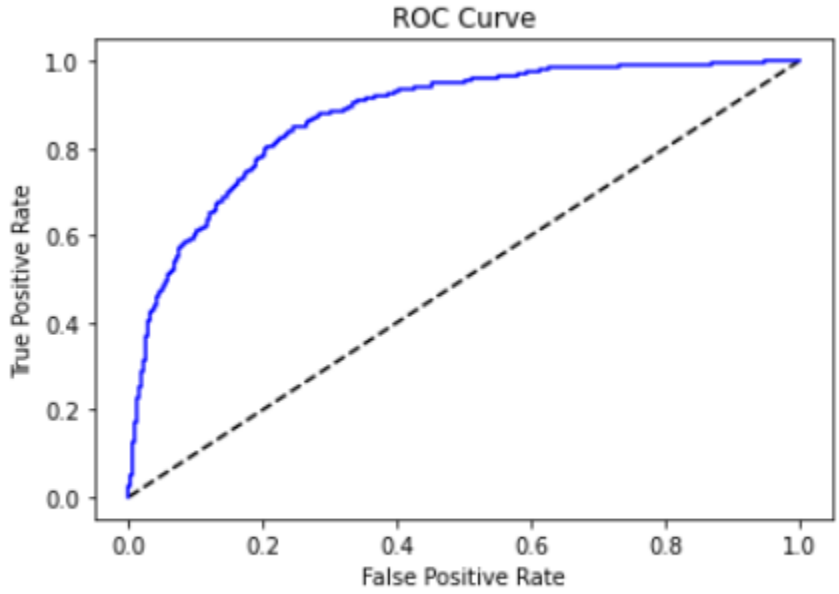
Confusion matrix:



Precision Recall Curve:



Area Under ROC Curve:




```
ROC_AUC:  0.8748315363881402
```

Fbeta:

```
0.8198726186591535
```

Confidence Intervals: (note 1 or positive classifications bring down the intervals drastically. Still a good score!)

```
95.0 confidence interval Recall: 57.1% and 60.7%
95.0 confidence interval Precision: 72.5% and 74.9%
```

Not a bad start. The RF model looks promising for trying to bet unders.

ii) - K-Nearest Neighbors

Grid search optimal n# of neighbors and respective score:

```
n_neighbors:  24
0.8700222328206803
```

Classification report after Inputting them in the model and then fitting the data

```
Classification Report
--------------------------------------------------------
              precision    recall  f1-score   support

           0       0.86      0.91      0.88      1060
           1       0.71      0.58      0.64       392

    accuracy                           0.82      1452
   macro avg       0.78      0.75      0.76      1452
weighted avg       0.82      0.82      0.82      1452
```

The above is very similar to the RF model.

Mean AUC score using cross validation:

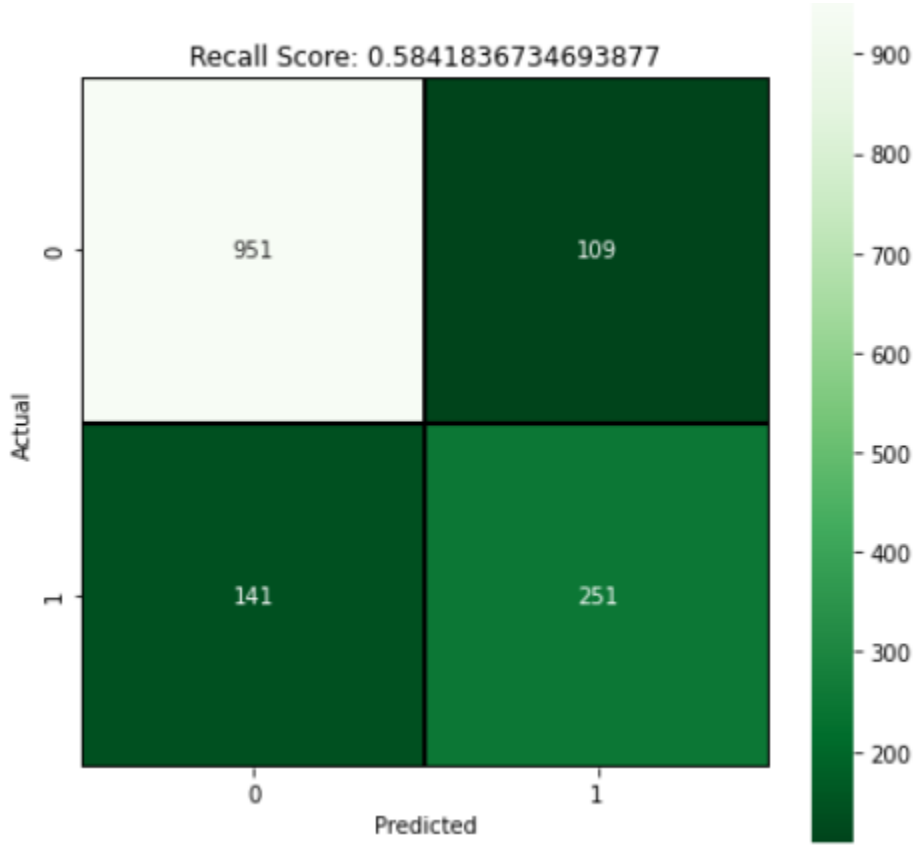
```
[0.8639779  0.88194553 0.87065266 0.86618986 0.86734521]
Mean AUC Score:  0.8700222328206803
```

The below classification report shows the precision, recall, and f-1 scores for 0 (Player grabbing 10 or less rebounds) vs 1 classifications (11 or above rebounds grabbed.) It looks like the model is accurate at predicting unders (0.91 recall on 0 classifications.) It is important to note that mean rebounds is 7.85, and our threshold is 10 and to take that into consideration accordingly when using the models in this project for betting purposes.

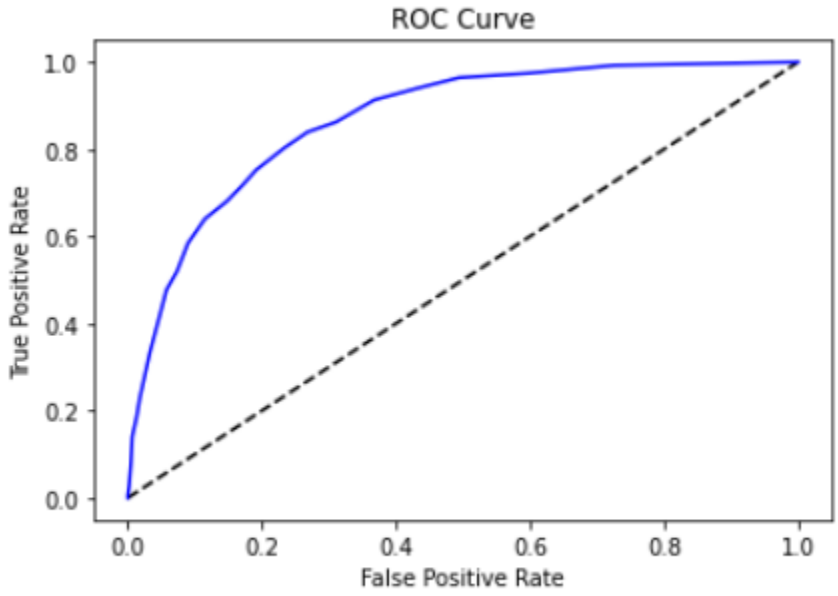
Classification Report

	precision	recall	f1-score	support
0	0.86	0.91	0.88	1060
1	0.70	0.59	0.64	392
accuracy			0.82	1452
macro avg	0.78	0.75	0.76	1452
weighted avg	0.82	0.82	0.82	1452

Confusion matrix:

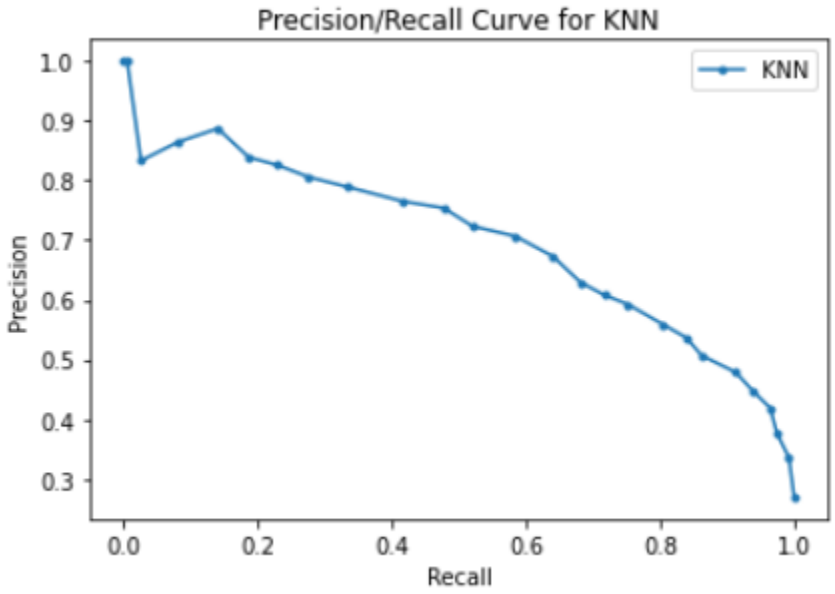


Area Under ROC Curve:



ROC_AUC: 0.8681772237196765

Precision Recall Curve:



Fbeta:

```
0.8195634654774715
```

Confidence Intervals: (note 1 or positive classifications bring down the intervals drastically. Still a good score!)

```
95.0 confidence interval Recall: 54.1% and 58.4%
95.0 confidence interval Precision: 70.4% and 74.6%
```

Worse, but not horrible. The RF model seems to be the better choice of now. It has a higher AUC score, and lower fbeta. Makes sense because the feature qualities fit better in decision trees.

iii) - Logistic Regression

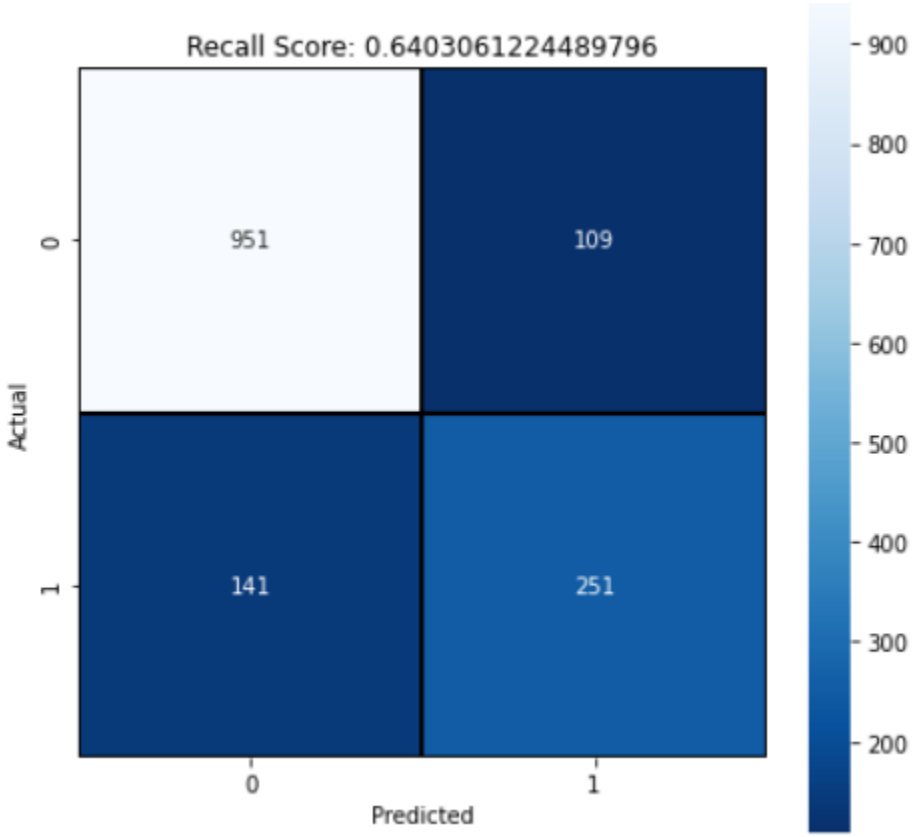
Grid search optimal C paremeter:

```
Tuned Logistic Regression Parameters: {'C': 0.05179474679231213}
Best Score: 0.8917851075458696
```

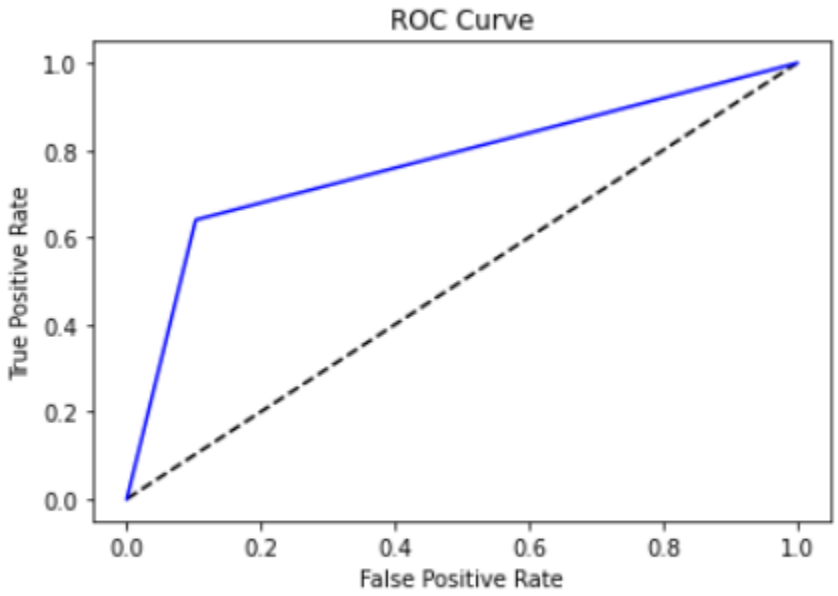
Mean AUC score using cross validation:

```
[0.88652775 0.90109234 0.89351906 0.89193708 0.88460336]
Mean AUC Score:  0.8915359186449479
```

Confusion matrix:



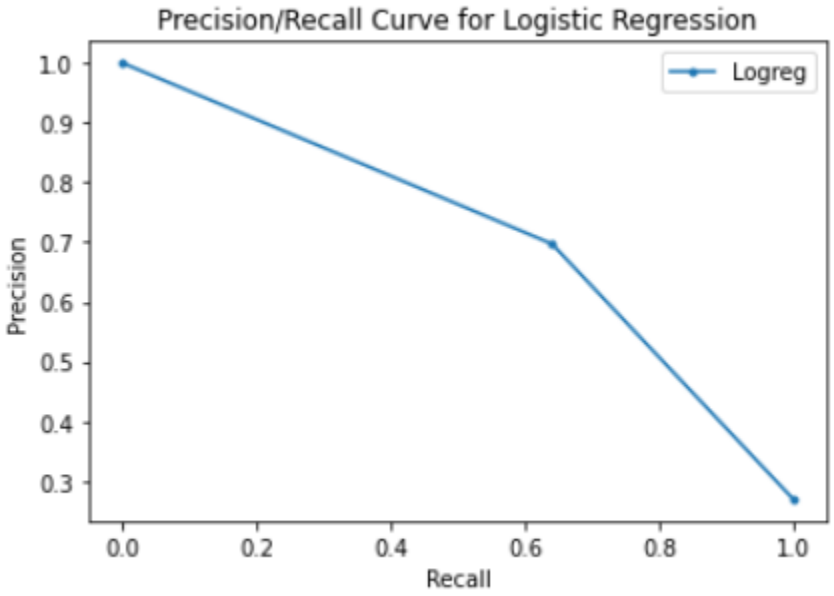
Area under ROC Curve:




```
ROC_AUC:  0.7687379668848672
```

Looks like the AUC in Logreg model is significantly lower than the RF and KNN models.

Precision Recall Curve:



Fbeta:

```
0.8267620855695731
```

Confidence Intervals: (note 1 or positive classifications bring down the intervals drastically. Still a good score!)

```
95.0 confidence interval Recall: 59.3% and 63.7%
95.0 confidence interval Precision: 71.7% and 75.6%
```

Worse as well, but higher confidence intervals.

Conclusion

Originally, I knew it would be quite hard to find promising results. Which is why I chose to pursue this project. Many people around the world are trying to crack the code, and though I achieved promising results at predicting Rebound under bets, nothing is valid until I put this model through the test. I am happy with the results, as it was difficult to obtain and wrangle all the data to get into the standards needed for modeling. I also was able to visually see certain thresholds that can be set as alerts, aside from the promising results achieved in the models. As I stated in the introduction, I was looking at this project to give me a clearer path forward and not for it to back up fixated ideas I had in my mind previously. Here are some cool alerts that can be taken from the EDA portion of the project:

- Bet overs after a player has averaged 17+ in the past 5 games
- Beware of betting unders during team win streaks
- Beware of “hard to rebound against teams” like Utah, and “easy to rebound against teams” like Brooklyn. Use this information to your advantage

I am going to test out the Random Forest Model on equal sized under bets on a rolling basis. The reason I am going to test it out on a rolling basis, and not for a fixed amount of games, is for the flexibility to adjust anything needed to be made to the model or strategy. If I had the spare time and resources, I would spend time creating and looking for more features, and adding in real-time updates such as player injury updates (IE: another feature with player position on home team, and player position on opposing team, and if they are injured or not.) I would also look into integrating the betting odds with the model to get a better understanding of implied winning odds. It is one thing to be 90% accurate at predicting unders, but what the unders pay off in certain situations is a big factor to profitability.

More limitations:

- Simply the natural variance in the sport
- Lack of in-game adjustments
- Bet sizing
- NBA Meta or rule changes
- Coaching strategies that are unknown prior to the game
- Other factors towards player wellbeing like amount of sleep, food intake quantity and quality, mood, etc.

For now however, the goal is to make a profit, and that can be made with 55% accuracy depending on the average odds, of course. I will put the results to use and continue to optimize.