Final Report:
Daily Fantasy Football Player Performance Predictor

Problem Statement

Sports betting was a 203 billion dollar industry in 2020, and it continues to grow every year. One subset of sports betting, Daily Fantasy Sports (DFS), is immensely popular. Due to the popularity though, it is difficult to pick players that perform well week-to-week, but also don't match what everyone else is picking. For this project, I set out to create a system that better predicts high scoring players for DraftKings Classic games.

Data Wrangling

Because the data I chose to use is mostly maintained by hobbyists (re: free), I expected this to be a much harder step than it was. In the end though, the data wrangling portion was very straightforward, with only minor issues. Thank goodness for dedicated sports stats people!

- **Problem 1:** Some of the salaries were missing. **Solution:** A lot of players that had missing salaries were low performers anyways, but due to the nature of football, with injuries and substitutions, there are oftentimes what are called "sleeper" players (players that very few, or no one, expected to do well for that week). So these had to be accounted for. For exploration, I tested 2 methods: substituting the mean player salary, and then minimizing their effects by giving them outrageous salaries (orders of magnitude larger). Neither helped in predicting sleepers, so I chose to minimize those players for the sake of training the models.
- **Problem 2:** The data that I thought would be useful was separated across a few different websites. **Solution:** Built a scraper that grabs the data off those sites and joins it altogether.

Exploratory Data Analysis

First, some notes about DraftKings Football DFS.

- While there are many different types of formats, 2 of the most played are Tournament and Double-ups (aka 50/50).
- In Tournament formats, only the top 10% (sometimes less) are paid out. So we are looking to score the absolute highest score possible. That means players picked are chosen because of their potential to get as many points as possible. For this format, we want to choose players that we expect to have breakout performances for the week.
- In the other formats (Double Ups and 50/50s), we only need to beat around ~50% of the field in order to be in the money (ITM). So instead of scoring as many points as possible, we want to choose players that have the highest minimum score. These would be players we expect with consistent mid- to high-tiers of performance.

Initially I set out to build a model that would predict lineups with the best chances of winning the Tournament styles (who doesn't want to win a $1 million?). I quickly found that seasonal data didn't really support that. Using data with patterns to predict outlier performances doesn't really make sense.

So I settled on building a model that would consistently produce good lineups for the 50/50 formats.

Here's what I found:

- Expensive players hardly ever justify their price tags
- Stacks are the most feasible in 50/50 formats
- Choose players based on the defense they play, not their own ability

In-depth Analysis

From my own anecdotal experience, as well as some quick internet searches, we know that winning lineups generally score in the 150s+ and the 180s+ for 50/50s and tournaments, respectively. So I wanted to see if I could isolate players that were likely to take my lineup scores into those ranges.

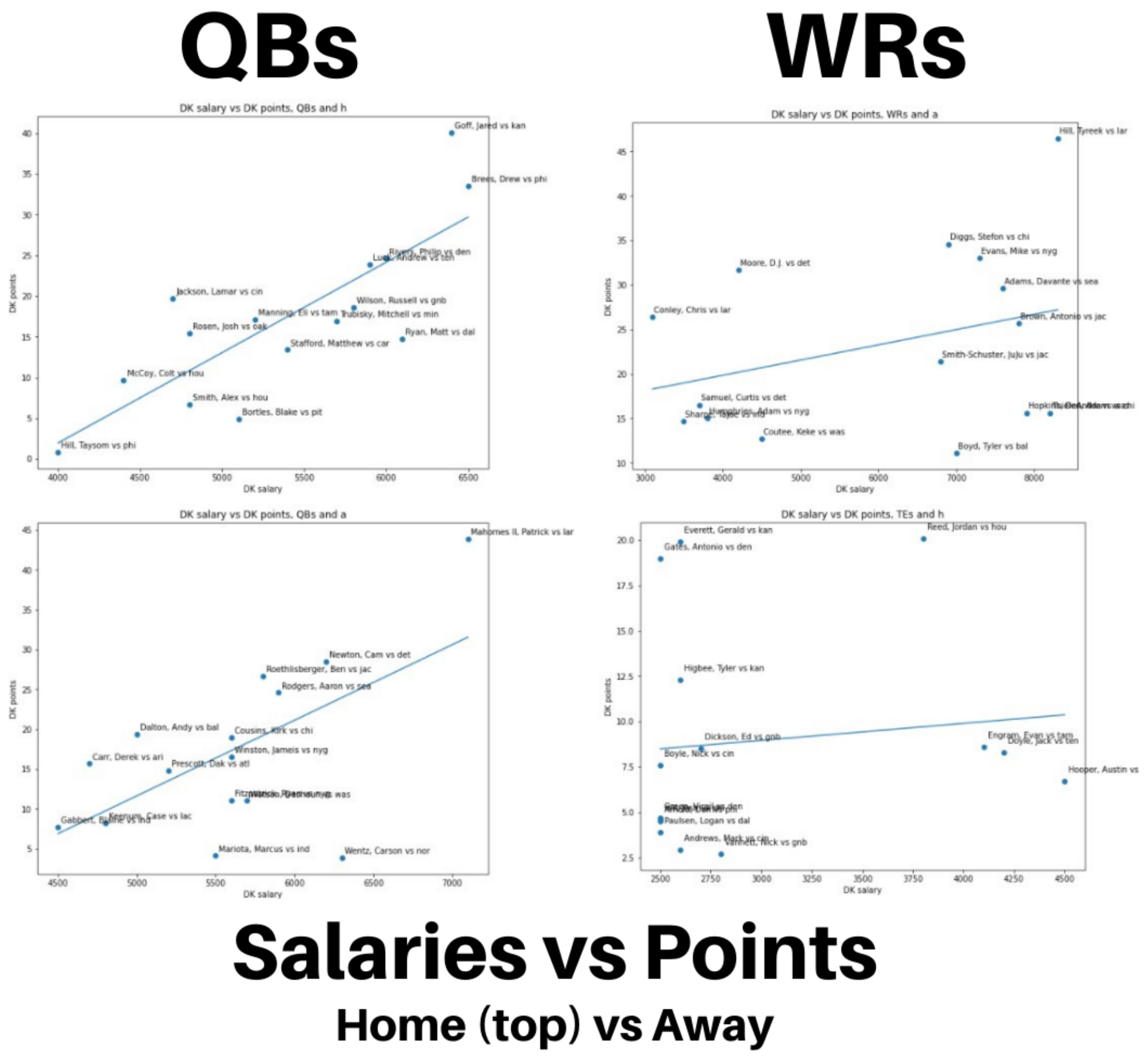
Some constraints about DFS lineups:

- Each lineup has a salary cap of $50,000, with each player's salary contributing to that cap.
- Each lineup has 9 positions that must be filled in order to complete the lineup.
    - 1 QB
    - 2 RB
    - 3 WR

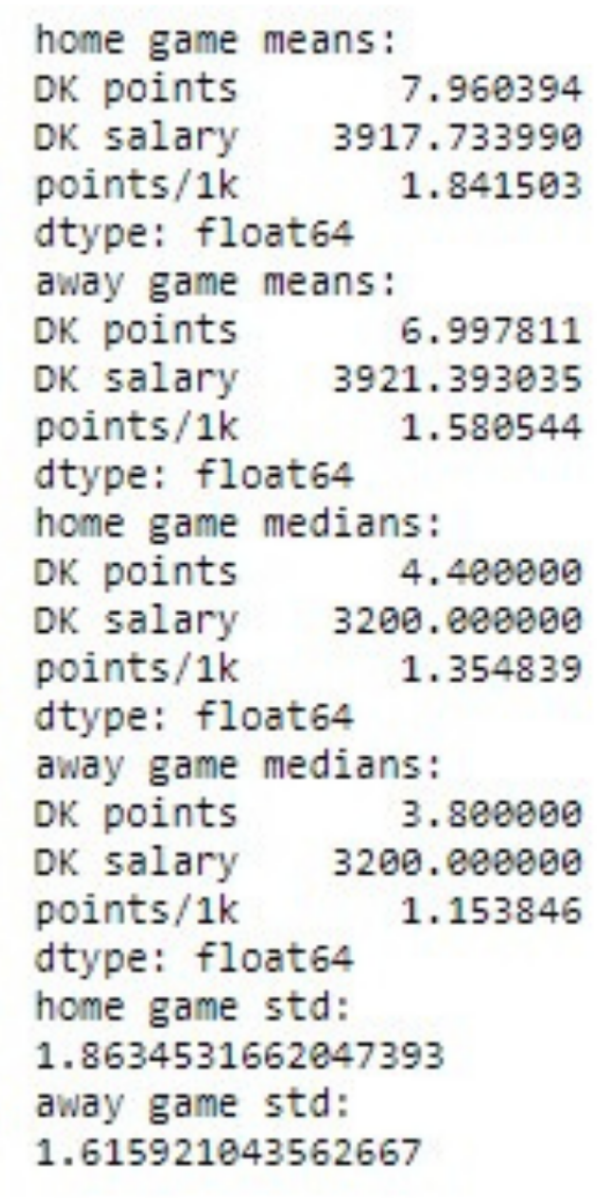
- 1 TE
- 1 Flex (WR or RB or TE)
- 1 Def

There are other constraints but these are the major ones we need to consider.

The data shows that, oftentimes, increased salary correlates with higher performances for QBs and RBs, demonstrated by the trend lines. Generally though, that pattern isn't reflected for TEs or WRs.



On average, players and defenses (although there is an interesting exception for defenses in week 11 of 2018) perform slightly better when playing at home.



```
home game means:
DK points        7.960394
DK salary     3917.733990
points/1k        1.841503
dtype: float64
away game means:
DK points        6.997811
DK salary     3921.393035
points/1k        1.580544
dtype: float64
home game medians:
DK points        4.400000
DK salary     3200.000000
points/1k        1.354839
dtype: float64
away game medians:
DK points        3.800000
DK salary     3200.000000
points/1k        1.153846
dtype: float64
home game std:
1.8634531662047393
away game std:
1.615921043562667
```

And the last thing I considered was a strategy known as “Stacking”, where several players on a team are added to a lineup, banking on the fact that the synergy (say, between a QB and his WR1) will add up to big points for any one lineup. Here’s what I found:

- When trying to choose a good (or even just decent) stack, it’s MUCH more likely to happen if you’re targeting mid-tier stacks.
- The most commonly occurring stacks are QBs with WRs.

There's more to this, but that gets discussed in the Takeaways.

Model Selection

This project was tackled with the SciKit Learn. Using their recommendation, I ended up testing 10 models. Out of those 10 models, 6 had good results that were fairly close together.

After optimizing and tuning those 6 models, 2 emerged as the winners. They had to be used in a specific way though. On their own, no model did a very good job of predicting high scoring players (the one with the most correct could only do it about 12% of the time.

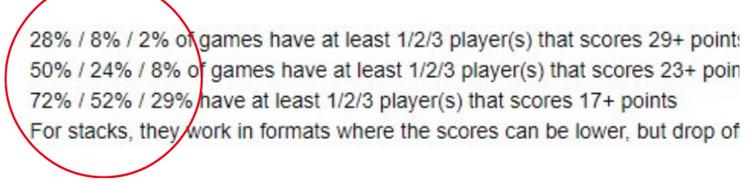
But what all of those models excelled at was filtering out the players likely to have low scores. So I took the model with the best results for picking "bad" players and filtered those out. Then I ran that new result through the model with the best rate for picking "good" players and the final result ended up being pretty good (if I do say so).

Before, picking players at random, you have roughly a 1-3% chance of choosing a high performing player at random, depending on the position. Using the models on their own, it's something along the lines of 7-12%, which is a really good improvement, but still not all that great in context.

Using the combination of models, filtering with Gradient Boost and then choosing players with AdaBoost improves chances to pick players scoring at least 17 points goes to 75%!

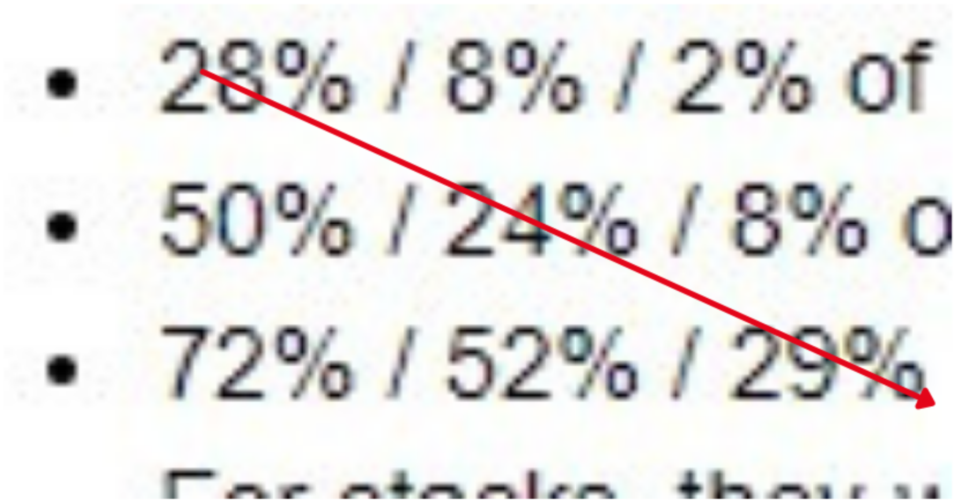
Takeaways

Before I get into the details here, let's revisit Stacking. The way I portrayed the results for stacks looks oddly like a confusion matrix:


28% / 8% / 2% of games have at least 1/2/3 player(s) that scores 29+ point
50% / 24% / 8% of games have at least 1/2/3 player(s) that scores 23+ poin
72% / 52% / 29% have at least 1/2/3 player(s) that scores 17+ points
For stacks, they work in formats where the scores can be lower, but drop of

And thinking about it that way made it way easier to think about. Obviously getting 3 players with over 29 points on the same team would be very hard to do (2% chance).

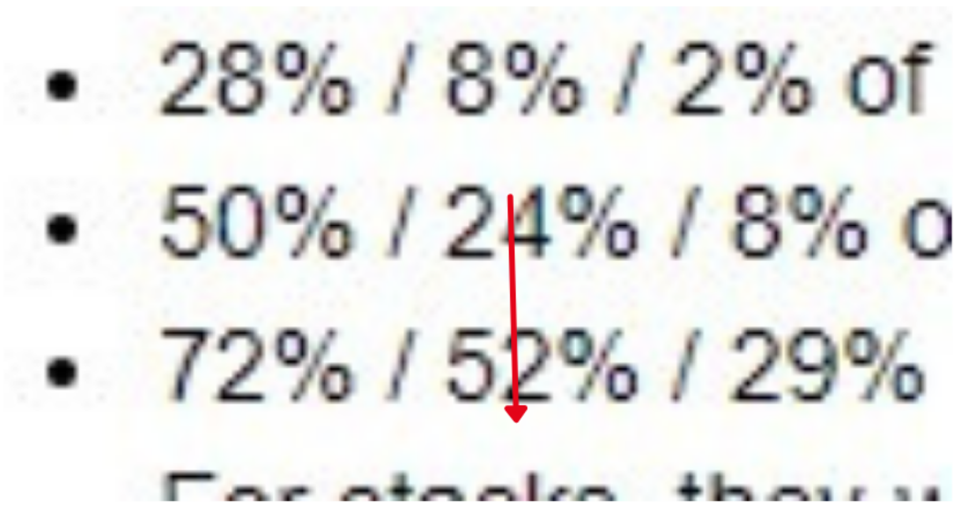
The next best, but also most plausible thing to shoot for would be...



- 28% / 8% / 2% of
- 50% / 24% / 8% o
- 72% / 52% / 29%

For stacks, they

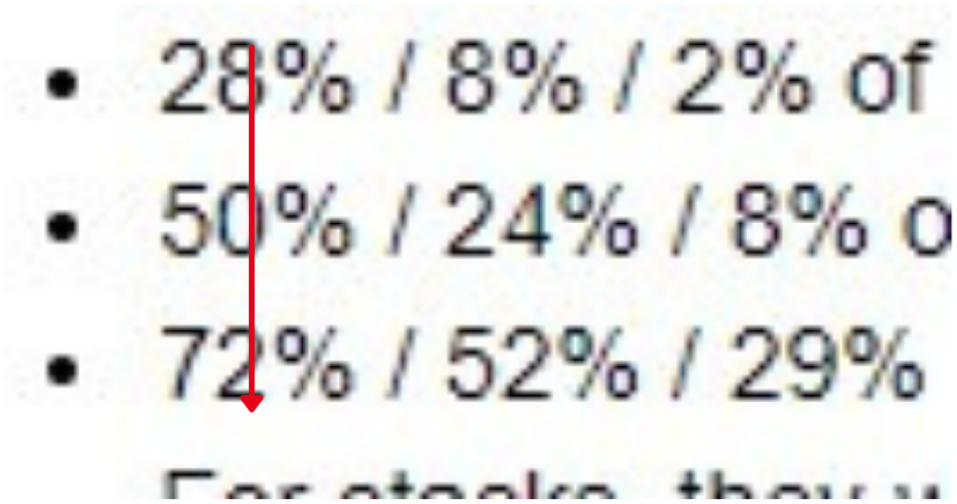
1 player with 29+, 2 players with 23+, and 3 players with 17+, which at minimum gives 126 points. But on closer examination, that comes out to about a 2% chance, and takes up extra roster spots, so what makes sense here is to chase the 3, 29+ pointers.



- 28% / 8% / 2% of
- 50% / 24% / 8% o
- 72% / 52% / 29%

For stacks, they

2 players with 23+ and 2 players with 17+, giving a minimum 80 pts. That’s about a 12.5% chance of happening, with a decent amount of points to boot.



- 28% / 8% / 2% of
- 50% / 24% / 8% o
- 72% / 52% / 29%

For stacks, they

And finally, what seems like the most likely to happen but also the least amount of points contributed, 1 player with 29+, 1 with 23+, and 1 with 17+ for 69 points. And as luck would have it, this is actually not the most likely to happen (comes out to about 10%). Lucky because now we don't have to entertain this as an option.

Now, the presumption here was picking players from teams at random. Thing is though: stacks aren't exactly random. Above, we noted that the most common stacks in the data are QBs with WRs. And those combinations make a lot of sense (QBs that throw a lot have a preferred target, which many times is a WR). Paired with the context of choosing good defensive matchups, it makes sense to try for stacks.

So, first takeaway: try stacks. If the models choose players that lend themselves well to stacks, then it makes sense to play them because in theory, they're more likely to occur if the model is working correctly.

Second takeaway, choose players based on defensive matchups, not innate ability, as the latter often leads to price inflation, which 1) we have shown doesn't always lead to good performances and 2) may break a lineup somewhere else.

Third takeaway, if you must spend up, spend on QBs and RBs, and not TEs, WRs, or Defs.

Last takeaway, try to pick players and defenses that are playing at home.

Future Research

The largest limitation was the free data I had access to. In the future, with some modicum of success with the models just built, I'd like to be able to use those proceeds to fund this project a little further and use paid data to really optimize results.

DFS also has just about every sport under the sun, so I'd like to use these ideas to take on sports like MLB and NBA (and their collegiate counterparts), sports that also have TONS of extra data as well as betting potential.