The first step of my approach was to examine and clean the data. This included:

* Eliminating unused columns
* Deleting rows with NaNs and verifying there was still enough useful data (all of 2019 had NaNs for example).
* Separating Data into train and test segment. Went with 2016-2018 for test, 2015 and earlier for training, and threw out the 2020 Covid year data for about a 25%/75% Test/Train ratio. Could have done a random train/test selection, but wanted to see if model held up projecting forward.
* Removing entries with full game spreads between 2.5 and -2.5 because those will have negligible correlation with the total.

Next, I used a single variable linear regression to predict 1st Half spreads just using full game closing lines. Same thing for Totals. This simplified down to H1 spreads being about 61% of full game spreads and H1 totals being 55% of full game totals. Better H1 models should explored with attention to key numbers, but this gave me a close approximation when compared to the H1 closing lines available in the dataset. So it was accurate enough for this exercise.

The linear regression gave me a mean and root mean squared error (or standard deviation) output for spreads and totals. Using this information, I made a function that gave me the win/push probabilities for any H1 spread and total by assuming a normal distribution. To calculate push probabilities, I took the cumulative density function between 2.5 and 3.5, as an example, if I was trying to estimate the chance of a push at 3. A major simplification in the interest of time was that I did not do anything special to account for increased push probabilities around key number (3, 7, totals 49 etc) .

To determine correlation % factors, I created a nested loop through all H1 spread/total combinations. At each combination, I calculated how often a favorite/over or dog/under occurred at each combination compared to if they were fully independent variables.

A correlation of 0% would be if the % occurrence of the combination of fav+over was just equal to the % of favs \* % of overs. For a Fav/Over combination with a Correlation of 1, the probability of favorite and over was equal to the probably of the lower likelihood event between favorite and over. Formula and Example below:

Corr % = (Fav+Over% - Fav% \* Over %)

------------------------------------------------

(Min(Fav%,Over%) - Fav% \* Over %)

For example if -7.5 favorites hit 50% of the time and 24.5 point overs hit 25% of the time, but I found that the fav/over parlay hit 20% compared to an expected 25%\*50% = 12.5%, then the correlation % would be: (20% - 12.5%)/(25%-12.5%) = 60%. Same process for dogs and unders.

I also generated values for whole spread and total numbers.

From this exercise, I found that the correlation of fav/over and dog/under were equal for each spread/total combination, and this makes sense. At mid to lower spreads and totals, I believe there were enough datapoints that I got good correlation factors at each combination and didn’t want to artificially smooth out the results by fitting to a model and lose accuracy near key numbers. At higher spreads and totals, where there aren’t really key numbers and where there were less datapoints in the dataset, I did use a linear regression based on spread and spread/total ratio to smooth out the noisy results. This smoothing process is shown with plots in the code (sorry for the “jet” color theme that some programmers don’t like). The final result was a Correlation % lookup table for any spread/total combination.

A major assumption for this effort was that the correlation % for a particular fav/over combination (say -7 & 28) is the same regardless of whether those are the -110 spreads and total numbers or longer/shorter shot occurrences. This can be examined further, but it does make some sense since the look up table just provides an adjust factor to the win/win probability. That adjustment can be the same whether the uncorrelated win/win probability is -110 or -1000.

Lastly, I combined it all together. The main function takes the full game closing spread/total, uses it to predict pricing/probability for the H1 spread and totals that were also input. Next, another subfunction looks up the correlation % for that spread and total combination. Lastly, I use the fact that for fair odds, the sum of the probabilities of each outcome times each outcome’s decimal odds = 1.

1 = win/win% \* parlay odds + 1\*push/push% + spread\_odds\*win-spread/push-total%

+ total\_odds\*win-total/push-spread%.

The win/win% is the spread -cover-probability \* total-cover-probability bumped up based on the correlation % by rearranging the Corr% formula presented earlier to solve for Fav&Over% (or dog and under). The spread and total odds come from the user input and the push probabilities come from the H1 model. The formula above is then rearranged to solve for the correlated ‘parlay odds’ term.

In the end, the productized function takes in the closing full game spread & total, the desired H1 spread & total and associated odds. It outputs the correlated parlay odds and an expected EV. For troubleshooting, I also calculate the parlay odds just from my H1 linear regression model without any correlation applied in order to see if the theoretical EV came from line inefficiency or correlation or both.

While the function works for any H1 spread and total, I did a simple check of its effectiveness by using the held back ‘test data’ set for 2016-2018. I calculated correlated parlay odds for all H1 spreads and totals in the set and generated an expected EV compared to an uncorrelated parlay with the individual odds provided. I summed all the projected positive EV entries and compared then against the profit generated by betting 1 Unit on all those +EV games. I then did the same thing, but pretended I was able to erroneously bet + or – 10 H1 Spread and 28 H1 total for every game at the same -110ish odds. This is obviously unrealistic, but it verifies the model works at different spreads/totals other than the closing H1 values. The projected EV and profit should be much higher because the base lines are inefficient and 10/28 have a decent correlation. Results for both shown below (and in the code). Expected EV calculated using training data models and the resulting profit using different test data line up well!

Future work would include refining the H1 model, and examining all the assumptions made in the correlation model.

BETTING H1 SPREAD AND TOTAL FROM TEST DATA SET FOR ALL +EV GAMES

Fav/Over sum EV of +EV parlays:

289.19

Fav/Over sum profit of +EV parlays:

127.44

Dog/Under sum EV of +EV parlays:

163.01

Dog/Under sum profit of +EV parlays:

199.70

Total sum EV of +EV parlays:

452.20

Total sum profit of +EV parlays:

327.14

BETTING + or - 10 SPREAD AND o/u 28 TOTAL FOR ALL +EV GAMES:

Fav/Over sum EV of +EV parlays:

398.17

Fav/Over sum profit of +EV parlays:

385.15

Dog/Under sum EV of +EV parlays:

540.85

Dog/Under sum profit of +EV parlays:

648.63

Total sum EV of +EV parlays:

939.02

Total sum profit of +EV parlays:

1033.78

Example single entry code run:

spread = -36.5

total = 51

H1\_spread = -22

H1\_spread\_price = 104

H1\_total = 28.5

H1\_total\_price = -110

([cor\_parlay\_am,EV])=parlay\_price(spread,total,H1\_spread,H1\_spread\_price,H1\_total,H1\_total\_price)

Model Decimal Odds to bet Fav:

1.98

Probability Fav covers:

0.489

Probability Spread Pushes:

0.033

Model Decimal Odds to bet Over:

2.32

Probability Over covers:

0.432

Probability Total Pushes:

0.000

Uncorrelated Parlay - Decimal:

3.89

Uncorrelated H1 Model Parlay - Decimal:

4.60

Correlated Parlay Price - Decimal:

2.80

Correlation Percent:

0.62

Earned Value:

0.39

Correlated Parlay Price - American:

180