CS 109 Challenge: An Analysis of Larry Sabato’s Crystal Ball

Walker Stewart

November 2021

**Background**

I have followed politics for years, but recently I have found that what I spend the most time reading and thinking about are elections, more precisely looking at polling, analysis, and predictions. Fundamentally, I believe the reason I am more interested in elections than any given issue that may galvanize Washington DC for a week or two stems from the enjoyment I get out of meeting and getting to know all kinds of different people across this country, and the decisions different people and groups of people make when voting is fascinating.

As a result, I wanted to apply what I have learned in CS 109 this quarter to election data. One of my favorite places to look for coverage of politics and elections is Larry Sabato’s Crystal Ball because of their frank and remarkably accurate coverage and predictions of elections. As a result, I thought it would be worthwhile to dive into the Crystal Ball’s election predictions and take a look at the outcomes of races depending on how they were rated.

**Data**

Using the Crystal Ball’s archives, I decided to look at all US House races from 2012-2020. There were a few reasons for this choice. First, because there are 435 House races each even year, there would be a large enough sample size to do worthwhile analysis. Second, the 2012-2020 window limits the challenge of dealing with changes in election laws and (mostly) redistricting. Finally, the more recent time-period accounts for the fact that predictions made further in the past reflect a different political atmosphere.

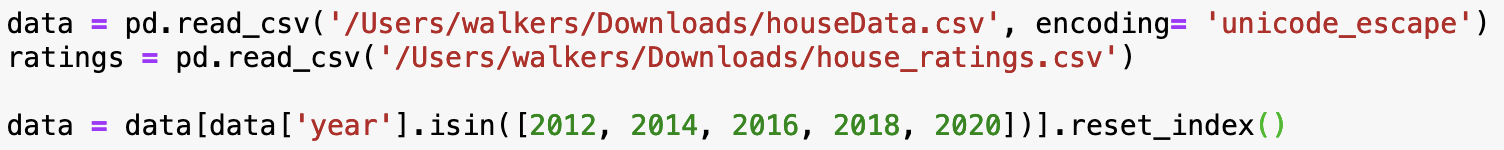
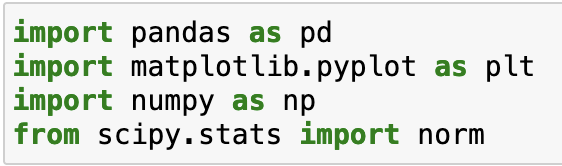
In the archive, I looked at the final ratings Crystal Ball released before an election. In this set of ratings, each race is assigned one of 6 ratings: Safe Democrat, Likely Democrat, Leans Democrat, Leans Republican, Likely Republican, or Safe Republican. I compiled the data manually, referencing the webpage with the ratings and creating a spreadsheet that matched the data (year, state, district, rating). In doing this, I reflected each of the 6 ratings numerically, with 1 representing Safe Democrat through 6 representing Safe Republican. Oftentimes, most of the safe races in either direction weren’t listed, I’ll discuss later how I dealt with that.



*Crystal Ball ratings before the 2012 elections*

As a point of comparison, I referenced the MIT Election Lab House Data, which contains the outcomes of all US House Races from 1976 until 2020 in the form of a downloadable csv with 21 columns (many of which weren’t necessary for me).

**Cleaning Data**

Once I had the MIT csv and the Crystal Ball csv, I loaded both of them into pandas dataframes in Python. In order to analyze the data, I needed to modify the MIT data. To preface this, my goal was to have a dataframe containing the two-party vote share in each congressional election from 2012-2020 to compare with the Crystal Ball predictions. The MIT data contains one row per candidate, I wanted to get to one row per race. This was a multi-step process.

*Getting set up and loading the data*

The first issue I needed to deal with was fusion ticket voting, a process allowed in 8 states but most commonly used in New York. With fusion ticket voting, a candidate can appear on the ballot multiple times under different party labels (for example the Republican and Conservative parties or the Democrat and Working Families parties). To address this, I created a new dataframe with only the fusion ticket races, grouped it by candidate and year and summed over the rows. I then checked a few candidates vote counts with what is listed elsewhere on the internet to ensure this process worked as I expected, then merged the new counts to my initial dataframe. Text

Description automatically generated

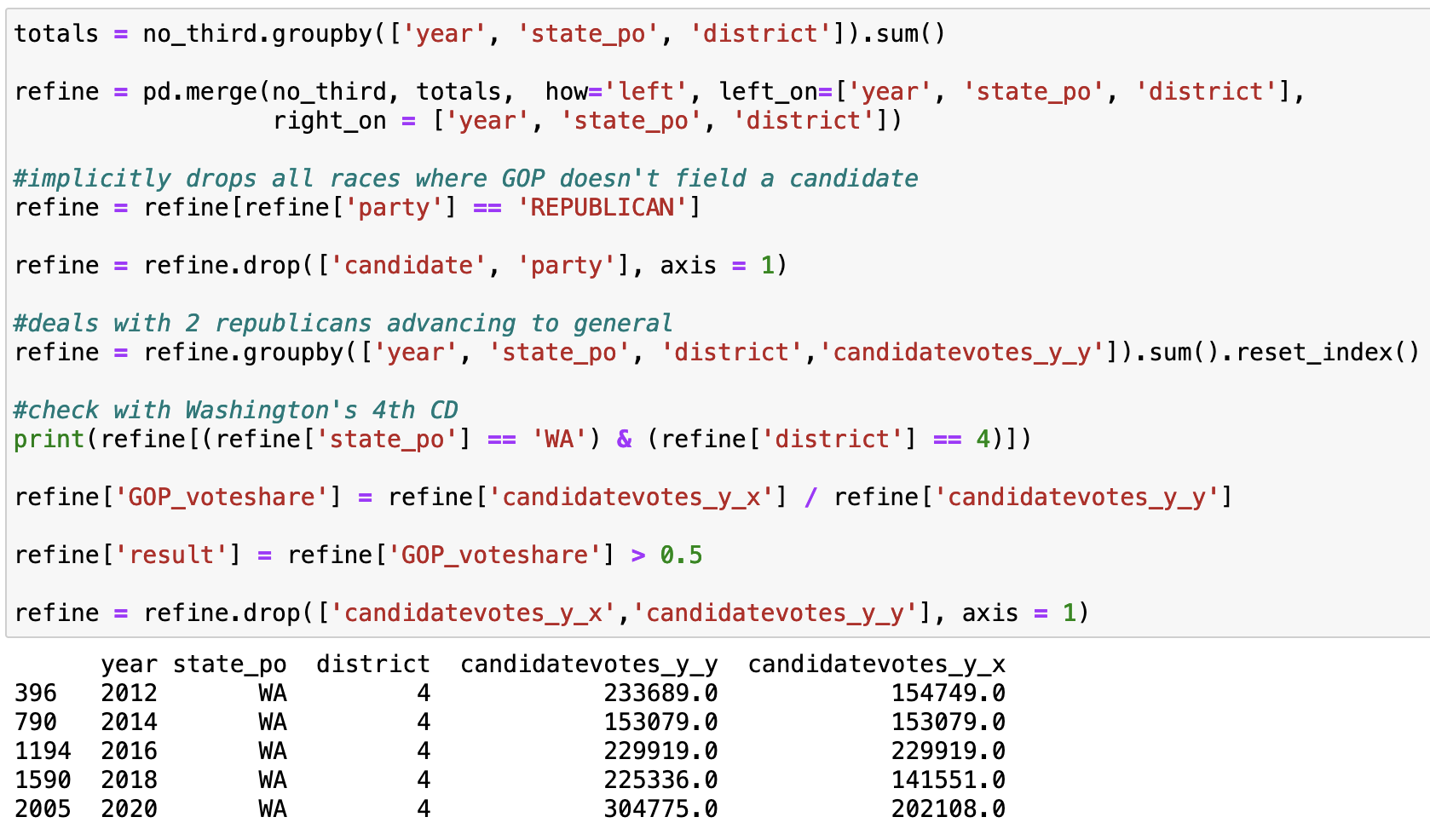
*Addressing the fusion ticket issue*

Next, because I cared about analyzing the two-party vote (no 3rd party candidate won a US House race from 2012-2020), and I had combined the fusion ticket votes, I could drop a lot of data I didn’t need, leaving me with a dataframe with only 6 columns.



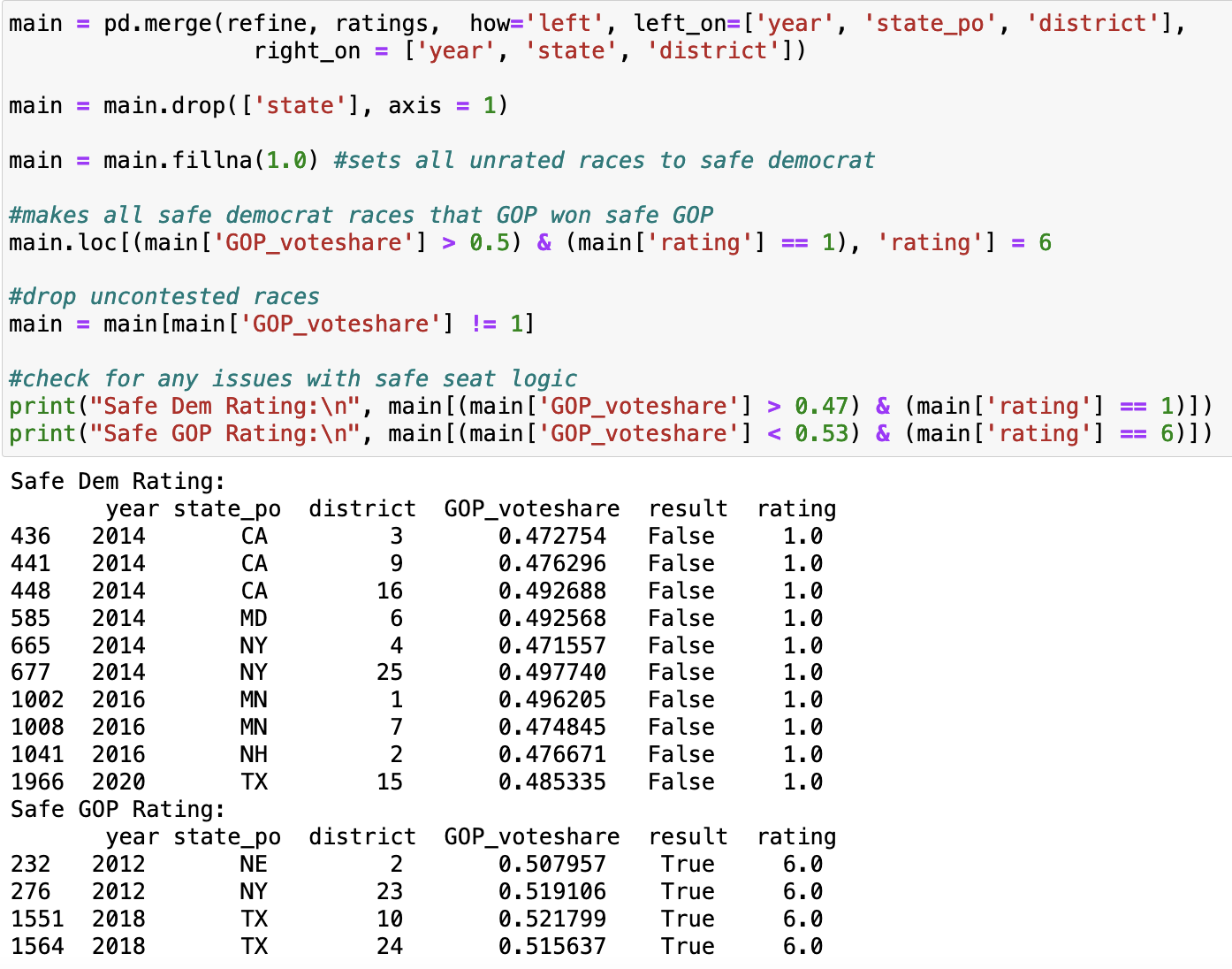
*Dropping superfluous data*

At this point, most districts had two rows, one for the Democrat candidate and one for the Republican. By the law of total probability (a deeper conversation for another time) Democrats win every race that the Republican does not exceed 50% of the two-party vote in, as a result I can create a column with the GOP share of the vote and that will give me all of the information I need. At this point I ran into another intricacy of state election laws, as California and Washington state have a top 2 primary system, which essentially means that in some cases a Republican could face another Republican in a general election.

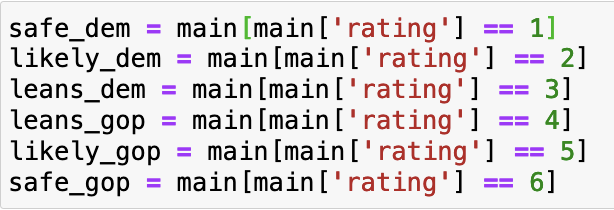


*Creating the GOP vote share column*

The next step was to merge the ratings data that I had gathered. As I mentioned earlier, Crystal Ball doesn’t list safe races on their pre-election analysis articles. To address this, I set every unrated race as safe for the party that won. To ensure I didn’t miss anything I individually checked every race that had a margin under 6 points, under the assumption that if a candidate pulls off an upset no one saw coming, it wouldn’t be by a wide margin. For brevity, I won’t dive into each of the listed races; in short my assumption was a good one.

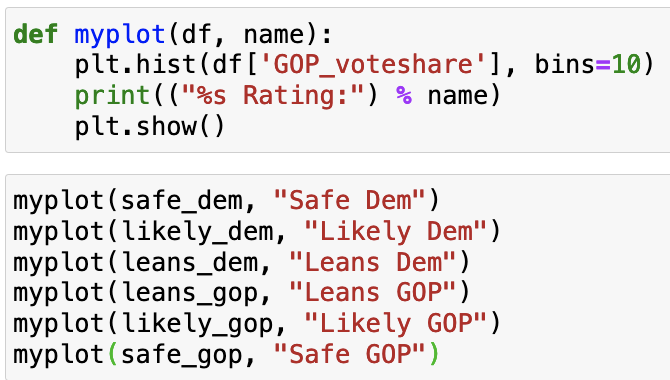


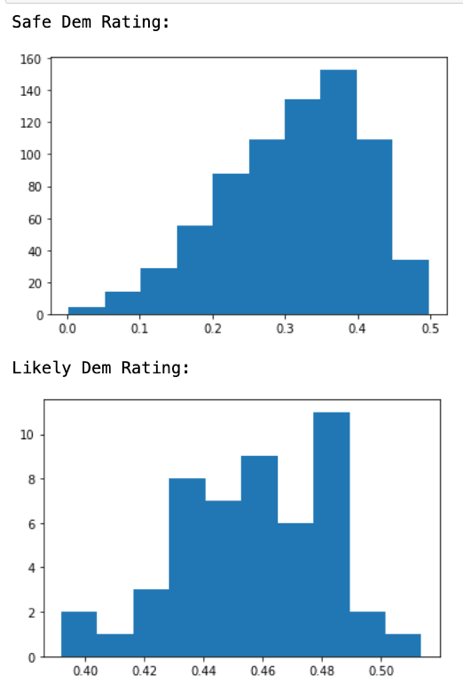
*Merging Ratings Data*

Finally, I created individual dataframes for each category of race. 

**Analysis**

As a first step, I wanted to take a look at the distribution of the votes across the different ratings. I wrote a short function that created histograms. Both the code and the results are below.



Chart, histogram

Description automatically generatedChart, histogram

Description automatically generated

*Histograms for the GOP share of the two-party vote (x-axis) by Crystal Ball rating*

One thing to note from these histograms is the significantly higher number of races that were [completely uncompetitive](https://gai.georgetown.edu/the-houses-competitiveness-problem-or-lack-thereof/). Another conversation outside the scope of this project. The next stage in my preliminary analysis was to look at the probability of a Republican victory given the Crystal Ball pre-election rating.

Text

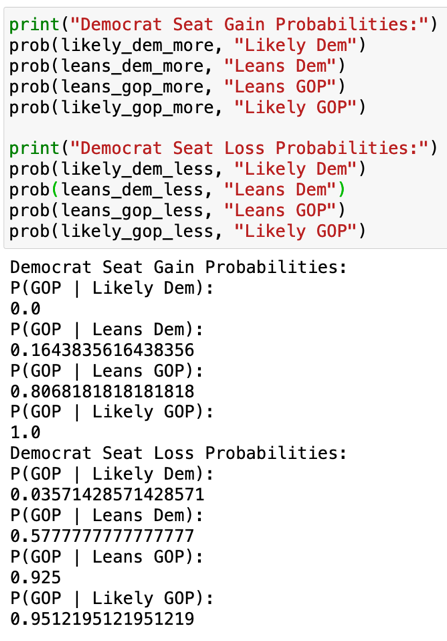
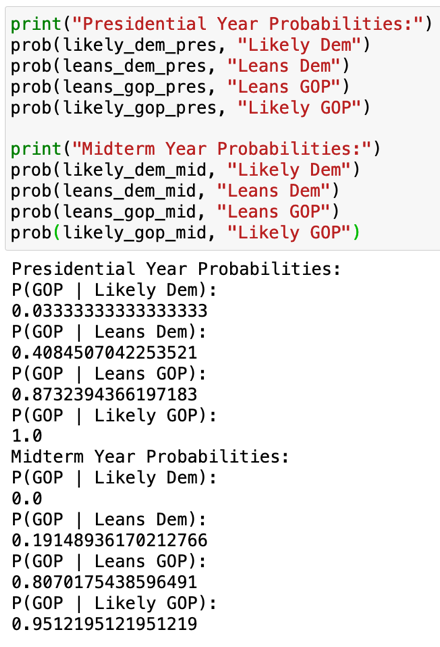
Description automatically generated

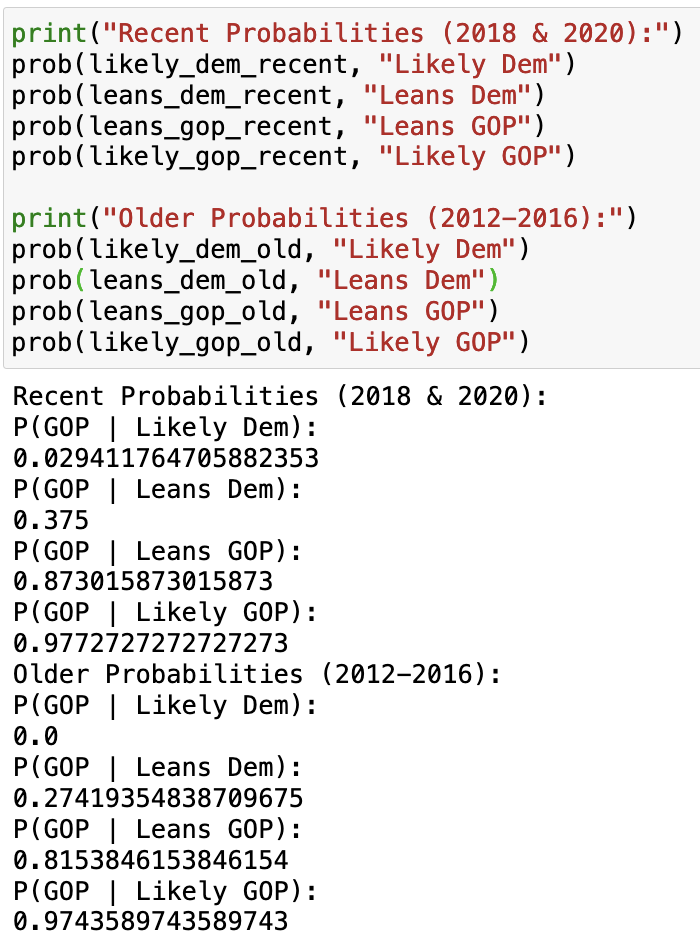
*Calculating GOP probability of victory for each of the 6 ratings*

For the races rated safe or likely for either party, the results are consistent at 100% and 98% respectively. What jumped out at me is the fact that Democrats won about 2/3 of the races that were leaning towards them, while Republicans won 5/6 of those races. I wanted to dive deeper and see why this was the case. Was Crystal Ball too conservative in changing a rating from lean GOP to likely GOP? Had they made errors earlier in the decade and corrected them? Was there a bias in years Democrats won seats compared to years Democrats lost seats? Was there a difference in Presidential years compared to midterms? I created dataframes for each of these cases and ran the probability function for each. I didn’t bother with the safe seats for each party because no matter how they were broken down the probability would remain the same.



*Setting the dataframes up (see note in reflections on this)*



Text

Description automatically generated

*The first four breakdowns I put together*

There were a lot of trends to see here, I chose to dive deeper into the recency category.

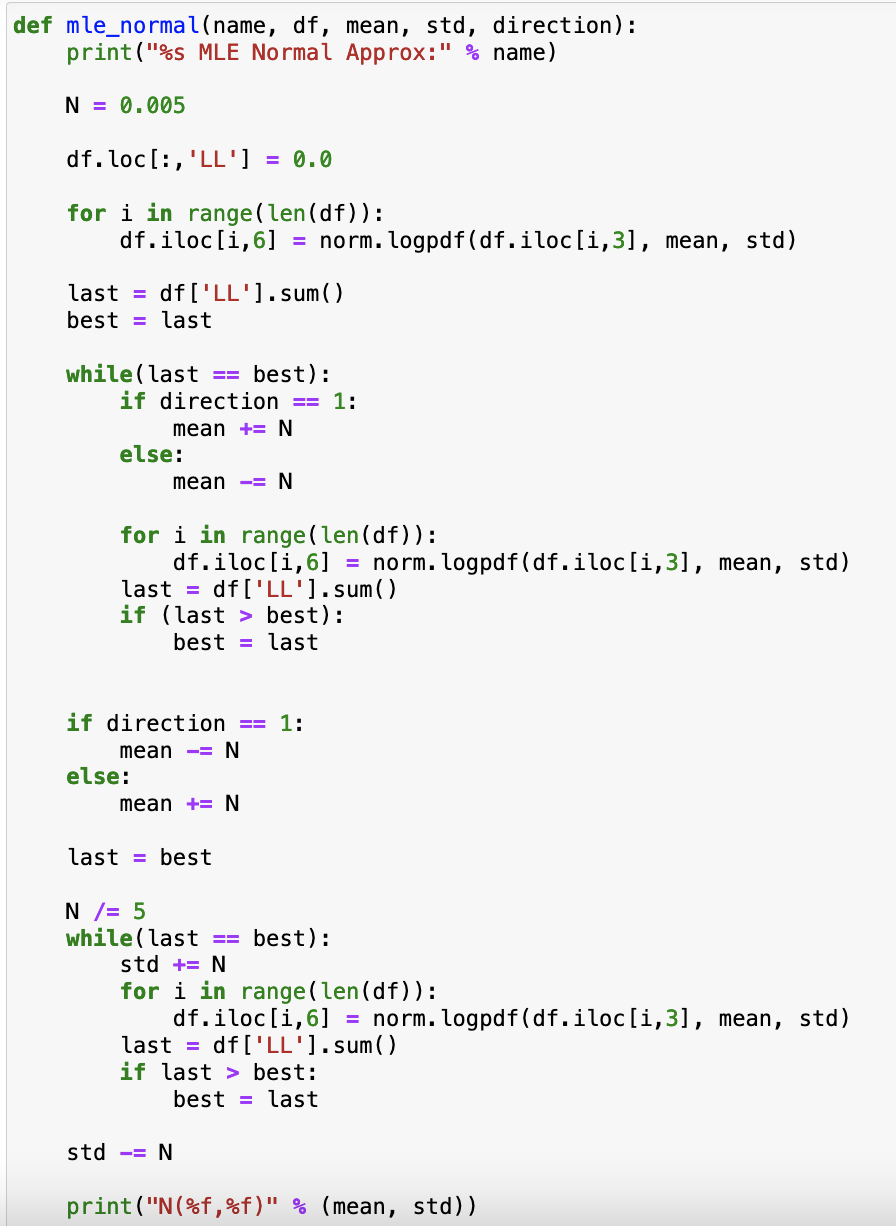
Text

Description automatically generated

*2020 data separated from the other years*

This data was quite interesting, because is shows that the Crystal Ball might have been a bit foggy in 2020, at least in their House analysis. From 2012-2018, seats rated as leaning towards one party or the other went in the direction of the lean around 80% of the time. In 2020, on the other hand, the GOP won 100% of races at least leaning in their direction, as well as 70% of the seats leaning towards Democrats. This unexpected outperformance accounts for the difference I mentioned above.

One observation I made when looking at the histograms for the races rated as leaning or likely is that they looked approximately normally distributed. Because of this, I decided to use MLE to see if I could derive the normal distributions.

Text

Description automatically generated

*Code I used to derive the normal distributions and the outcomes for each category*

To analyze the accuracy of these gaussians, I wanted to compare the gaussian prediction for the chance of a GOP win to the observed outcomes.

Text

Description automatically generated

*Testing the accuracy of the gaussians*

While the approximations were quite accurate, I would posit that their predictive power is minimal given the relatively small sample size and the lack of differentiation between testing and training data.

**Reflection**

Areas for Further Research

I found this project fascinating and I am certainly going to play around with this data more in the future. One thing that may be interesting would be to include ratings from other prognosticators, and ratings from other points in the campaign.

Areas of Potential Improvement

When I was cleaning the data, I had a clear goal in mind and broke the process down into individual tasks, but once I began analyzing the data I didn’t have any specific direction I wanted to go in besides look around and see if there is anything cool I could find. As a result, the code where I create 30 different dataframes and put all of them in a function could have been done much more efficiently had I used a dictionary of dataframes.

Additionally, while I caught and fixed a few obvious errors (a Virginia candidate was listed as having been on a fusion ticket despite the practice not being legal in Virginia) in the election data it is plausible that other errors persisted and impacted the accuracy of my analysis. Furthermore, I entered all 400 Crystal ball ratings by hand in Excel, I could have mistyped a district number without knowing it as well.

**Works Cited**

Larry Sabato’s Crystal Ball

MIT Election Lab

Wikipedia Election Results

Ballotpedia Election Results

Pandas Docs

Scipy Docs