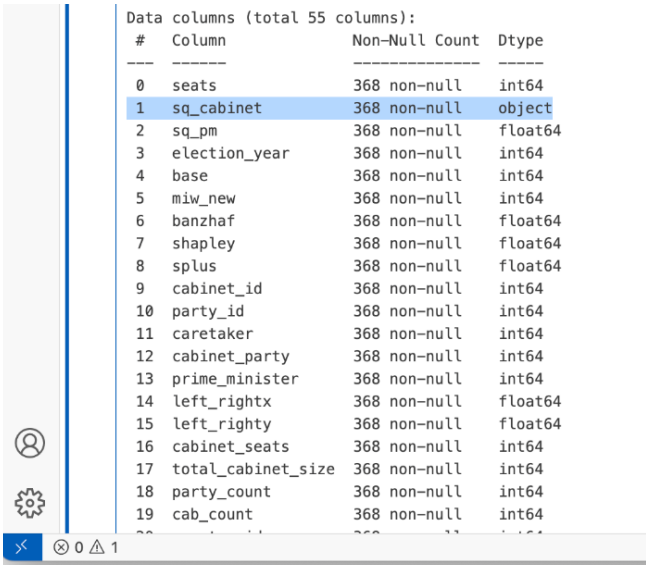
- all steps #s correspond with single-line comments in the cells of the jupyter notebook/code.
- imported all libraries in step 1.

Explanation for data framing/how I dealt with missingness:

After opening the csv file, I printed a .info() table to get an idea for how much missingness was in the dataframe (steps 1-3). I noticed that there was some missingness in the data; specifically, `coalition_total` was missing for 424 of the total 657 observations. I dropped `coalition_total`—my reasoning for dropping this IV was that the measurement was missing for more than 50% of the observations, and I was trying to **balance** having to drop too many individual rows with NaN values through .dropna **with** dropping an IV itself that had too much missingness. I didn't want to impute this variable, but I may have been able to google it for each party in the dataset if my time was actually infinite during finals week. I reasoned that if whoever compiled this data set made sure that most of the observations didn't have it or the European democracy didn't track it and it wasn't google-able, `coalition_total` was not a significant IV to begin with (step 4).

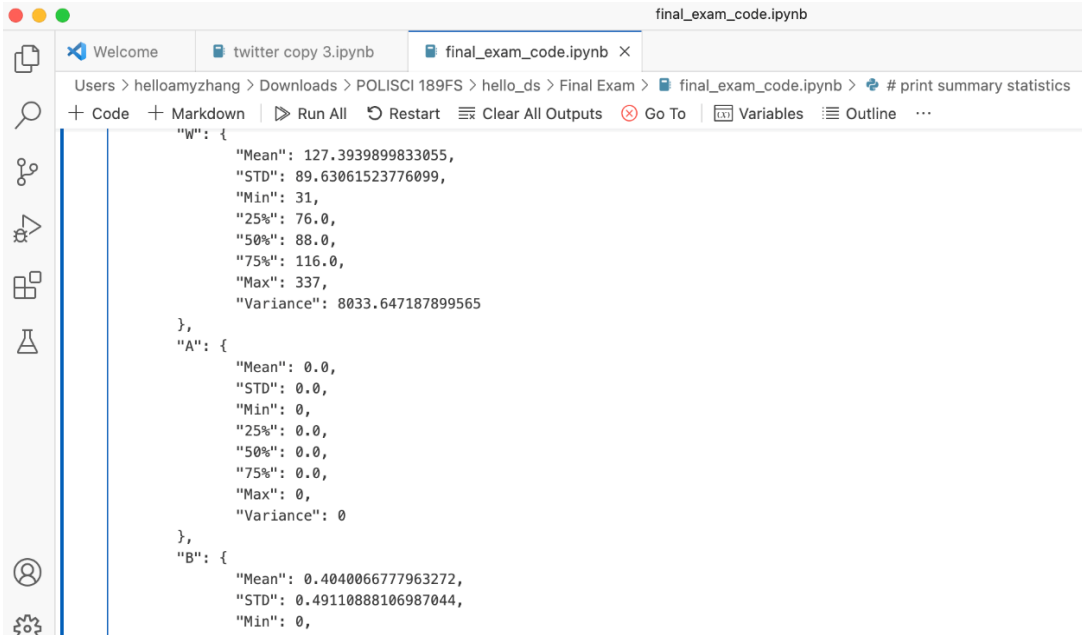
I also dropped `party_name`, `party`, `country`, `cabinet_name`, `party_name_english`, `election_date`, and `start_date` because they were object/string data types, and I wouldn't be able to feed those into a decision tree later. After dropping these IVs, I ran .dropna() on the data frame to get rid of rows with any remaining missingness or null values, and went from 657 to 599 observations. I felt that almost six hundred observations was enough data points to train and test a few models.



```
Data columns (total 55 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   seats               368 non-null    int64
 1   sq_cabinet          368 non-null    object
 2   sq_pm               368 non-null    float64
 3   election_year       368 non-null    int64
 4   base                368 non-null    int64
 5   miw_new             368 non-null    int64
 6   banzhaf             368 non-null    float64
 7   shapley             368 non-null    float64
 8   splus               368 non-null    float64
 9   cabinet_id          368 non-null    int64
 10  party_id            368 non-null    int64
 11  caretaker           368 non-null    int64
 12  cabinet_party       368 non-null    int64
 13  prime_minister      368 non-null    int64
 14  left_rightx         368 non-null    float64
 15  left_righty         368 non-null    float64
 16  cabinet_seats       368 non-null    int64
 17  total_cabinet_size  368 non-null    int64
 18  party_count         368 non-null    int64
 19  cab_count           368 non-null    int64
```

At this point, I had 55 potential IVs to look through; I had eliminated 8 IVs in previous steps. I printed summary statistics for 54 of the IVs (the remaining 55 sans `sq_cabinet` because my interpreter was treating it as an object type instead of an int type even though it looked to be a column of ones and zeroes—I just popped it out so my summary statistics cell could run in step 5).

After looking through the summary statistics, I noticed this:



final_exam_code.ipynb

Welcome | twitter copy 3.ipynb | final_exam_code.ipynb

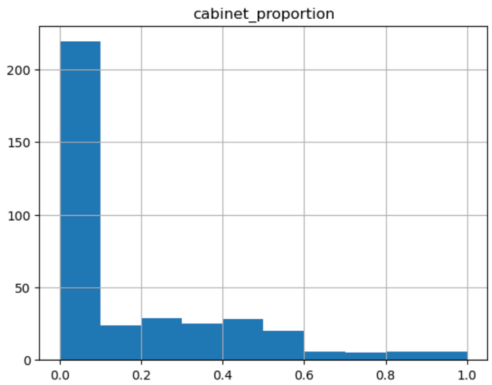
Users > helloamyzhang > Downloads > POLISCI 189FS > hello_ds > Final Exam > final_exam_code.ipynb > # print summary statistics

+ Code | + Markdown | Run All | Restart | Clear All Outputs | Go To | Variables | Outline | ...


```
        "W": {
            "Mean": 127.3939899833055,
            "STD": 89.63061523776099,
            "Min": 31,
            "25%": 76.0,
            "50%": 88.0,
            "75%": 116.0,
            "Max": 337,
            "Variance": 8033.647187899565
        },
        "A": {
            "Mean": 0.0,
            "STD": 0.0,
            "Min": 0,
            "25%": 0.0,
            "50%": 0.0,
            "75%": 0.0,
            "Max": 0,
            "Variance": 0
        },
        "B": {
            "Mean": 0.4040066777963272,
            "STD": 0.49110888106987044,
            "Min": 0,
```

Because `A` had 0 variance, a mean of 0.0, quartiles of 0.0, and seemed to be a column of zeros, I dropped it. `A` was had no variance, and we want all of our predictors in a polynomial regression to have variance. After dropping `A`, I had 53 IVs (step 6).

In step 7, I ran histograms on `cabinet_proportion` and all of the IVs. I noticed that the DV `cabinet_proportion` was significantly right-skewed and clumped around zero; i.e., the DV wasn't Gaussian, but zero-inflated. I also noticed this property in some of the IVs. I tried running a seaborn pairplot in step 8, but the cell was taking hours/probably eventually days to run, so I commented it out in the jupyter notebook. Instead, I ran the alternative—a `scatter_matrix` from the pandas library. It wasn't super helpful; I didn't notice anything amiss. I histogrammed the DV again (just to make sure I didn't do it wrong in step 7), but alas, it was still zero-inflated. I resigned myself to not being able to run an OLS on my DV. I also initialized a variable `y` as my DV (the cabinet_proportion part of my dataframe in step 7.



While running the `scatter_matrix`, I got a warning from anaconda3 because of the line `scatter_matrix(updated_data, alpha=0.2, figsize=(25, 25), diagonal='kde')` in my step 7 cell. If you run my code; you'll get this warning too — the `scatter_matrix` still outputs, but you'll have to manually click 'run' on the next cell for the rest of my code to run, or you can comment the problematic line out.

```
LinAlgError: The data appears to lie in a lower-dimensional subspace of the space in which it is expressed. This has resulted in a singular data covariance matrix, which cannot be treated using the algorithms implemented in `gaussian_kde`. Consider performing principal component analysis / dimensionality reduction and using `gaussian_kde` with the transformed data.
```

I also established my baseline expectations for my future models (step 9) by running the standard deviation of `y`, also known as `cabinet_proportion`. The sample variance of `y` was 0.059, and this became my baseline for MSE in any future model I'd run in this assignment. The mean of `y` was 0.1666 and standard deviation of `y` was 0.243; these figures respectively became the a) average value that should be predicted for cabinet_proportion in a model and b) ideal model RMSE. Because MSE was the most convenient model performance indicator to check (in the models I'd run for this assignment), I only used the 0.059 baseline from this cell, and overlooked `std` and `mean` of `y`.

In step 10, I ran VIF on all of my remaining IVs, and printed them into a table. I noticed that some of the VIF values were `NaN` or `inf`—the latter made sense to me later, because they're dummy variables for a specific country and as a result, are probably highly (multi)collinear with other variables in the set of IVs. Because VIF can help one identify multicollinearity by estimating the amount of variance in one variable that can be explained by others, I was alarmed by the high VIF values in this dataset. Specifically, `left_rightx` and `left_righty` both had VIFs greater than 25. VIFs of 25 are incredibly high, considering that the ISLR textbook suggests that if a variable's VIF is equal to or greater than 5 or 10, it is multicollinear in your dataset. I decided to run VIF again (step 11), but instead of printing the VIF values into a table, I only printed the variables that had a VIF above a threshold of 10. I got a list of 30 variables with VIF > 10; the variables were `['left_righty', 'seats', 'base', 'banzhaf', 'shapley', 'splus', 'left_rightx', 'party_count', 'country_id', 'seats_share', 'enpp', 'bicameral', 'seats_total', 'miw_proportion', 'seats_proportion', 'W', 'B', 'C', 'D', 'E', 'country_dummy1', 'country_dummy2', 'country_dummy3', 'country_dummy4', 'country_dummy5', 'country_dummy6', 'country_dummy8', 'country_dummy10',`

`'country_dummy11', 'country_dummy13']`. I didn't want to drop all 30 variables yet, so I decided to run a correlation matrix (step 12), even though correlation matrices can't identify multicollinearity.

Through the correlation matrix, I found that left_rightx and left_righty were correlated at `0.9661`, which was quite alarming. I also knew from my .info() table in step 3 that `left_righty` was missing for 289 of the original 657 observations, while `left_rightx` was only missing for 52 of the original 657 observations. Because `left_righty` was too highly collinear with `left_rightx` for observations that had both measurements, and because `left_righty` had much more missingness in comparison, I decided to drop `left_righty` from my remaining IVs. In hindsight, it may have been more ideal to combine `left_rightx` and `left_righty` into a latent variable through PCA after imputing for `left_righty`, especially since according to your Slack message, they are the x and y axes of a political space of left/right and libertarian/authoritarian. Through the correlation matrix, I also noticed that `base` and `seats` were perfectly correlated at 1.0 (they were identical columns), so I dropped `base`. I initialized my remaining 50 IVs in `ind_vars` and my `cabinet_proportion` as my `y` (step 14).

At this point, I was overwhelmed by the amount of (multi)collinearity among my 50 remaining IVs/by the list of correlation pairs in my set of IVs, and was inspired by Dr. Banks' recent lecture about network models, so I ran network diagrams (step 15) based upon the correlation pairs I identified earlier in my correlation matrix. I imported the `networkx` package, and most of the code I had to run to make these network diagrams happen came from https://stackoverflow.com/questions/56717750/plotting-similarity-matrix-using-networkx. Because it's not possible to truly visualize multicollinearity among variables through a network, I probably spent too much on this. However, these diagrams guided my choices for PCA later on.

Diagram 1:

- The correlation matrix pairs in this network were pairs of IVs (from the 50 remaining ones from the original data set) whose correlation was greater than 0.70.
- The closer two nodes— each node is an IV—are to each other, the higher their correlation. If an edge doesn't exist between two nodes, there is no correlation between them above the threshold of 0.70

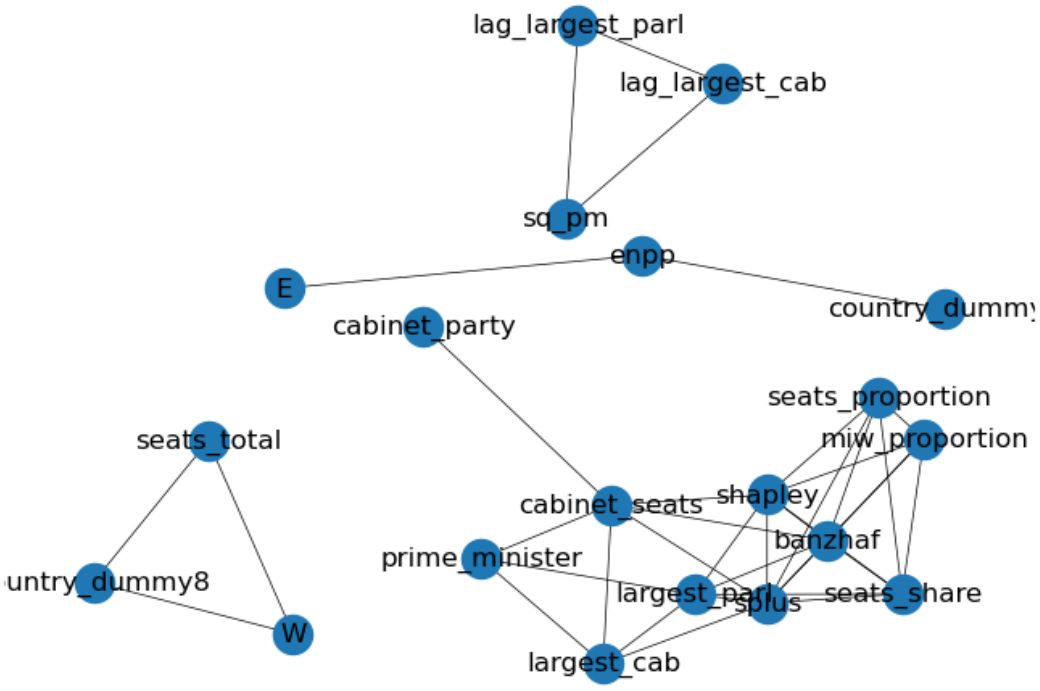
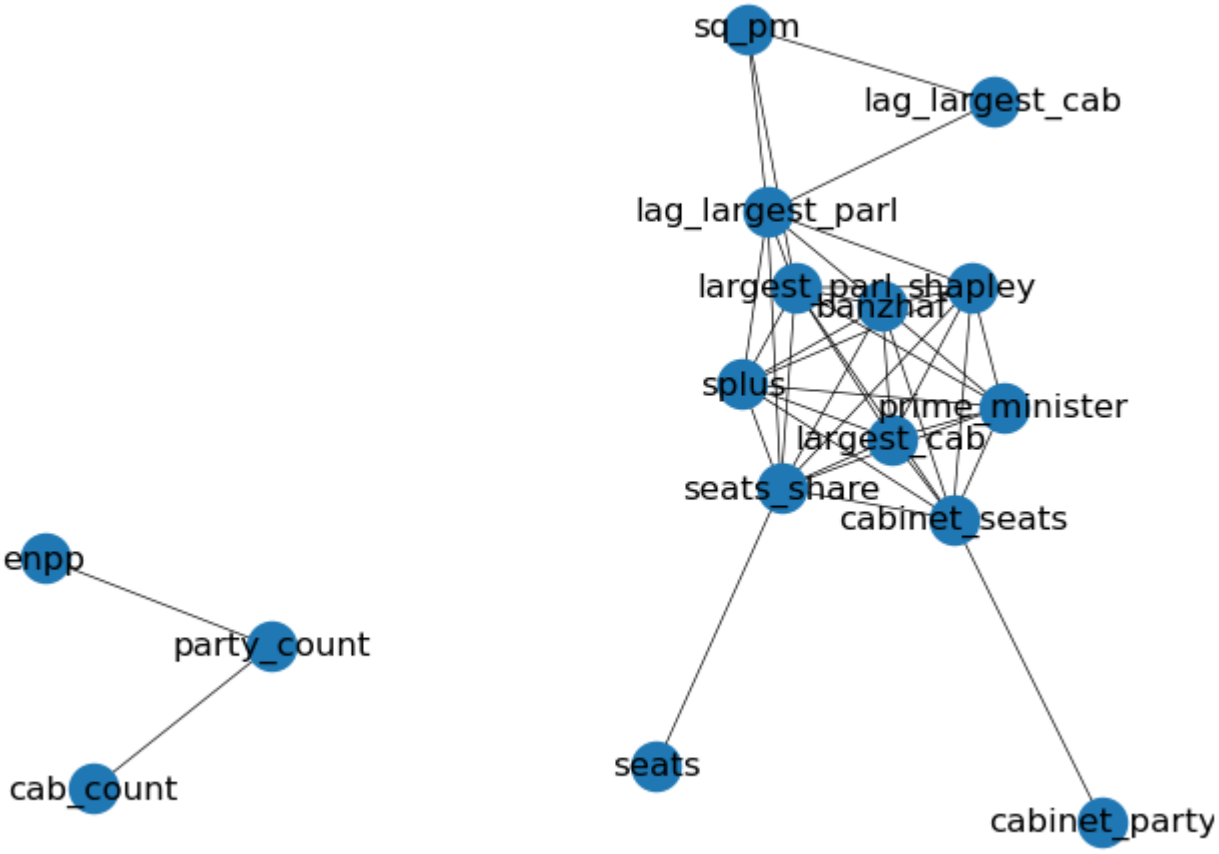


Diagram 2:

- The correlation matrix pairs in this network were pairs of IVs whose correlation was greater than 0.60.
- The closer two nodes—each node is an IV—are to each other, the higher their correlation. If an edge doesn't exist between two nodes, there is no correlation between them above the threshold of 0.60



Out of curiosity, I ran a third network diagram where I created an edge for every correlation stronger than 0.10 between all possible pairs of my 50 IVs, but it was unhelpful/too messy, and I couldn't actually identify some of the nodes/IVs. The third network diagram did not make it to this write-up.

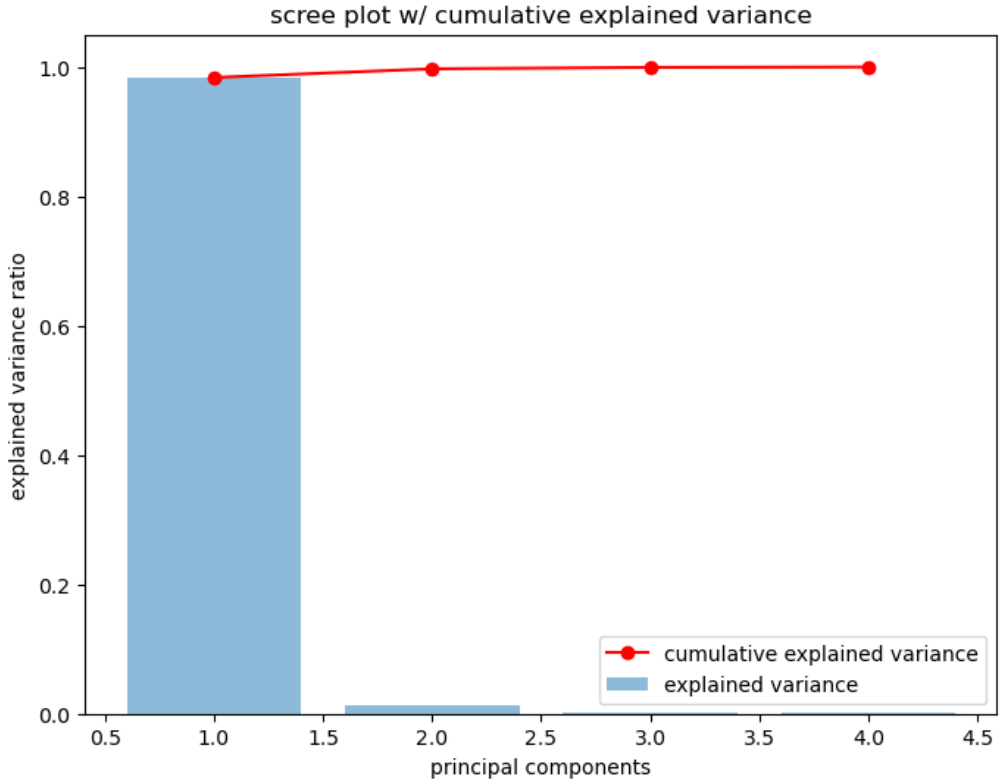
In Diagram 1, I noticed that there four connected components/islands, and in the connected component with the most nodes (in the lower right), the measures of pivotality that you had mentioned in Slack—`splus`, `shapley`, `banzhaf`, and `miw_proportion`—were highly correlated with each other, and to several other variables in the data set; in short, there was a high concentration of edges coming between the pivotality measures. I thought about dropping three of the four total pivotality measures/IVs and only keeping one of them, but was indecisive about which would be best based upon my limited theory. Your paper titled *Bargaining Strength and the Formation of Coalition Governments* "demonstrated that MIW's are in fact important in explaining coalition outcomes in parliamentary democracies," but your Slack message said that `splus` was "magic" and "incorporates future expectations of what one's shapley value will be if there's a new election." I couldn't even choose between `miw_proportion` and `splus` vs. `shapley`; as a result, I ended up throwing `splus`, `shapley`, `banzhaf`, and `miw_proportion` together into a PCA in step 16. I did a PCA attempt where I also threw in `miw_new` along with the four aforementioned IVs, but the coefficient on that linear combination/latent variable for `miw_new` was low (less than 0.10), so I took it out. I do not have a theory for why `miw_new` did not contribute very much to the latent variable.

In Diagram 2, there were 2 connected components, and I debated throwing each connected component as a set of IVs into a PCA (on the basis the IVs in the connected component were correlated/collinear with each other), but thought against it (because it would be too complex and difficult to explain each principal component in model interpretation, and I had no strong theory for doing such a PCA). I felt like I didn't understand the theory behind most of the IVs in this data set, and was nervous about trying to explain their relationships/the linear combinations of them if I ran more PCAs and created more latent variables. Because of this, I decided to run regularization/lasso later in order to feature-select the non-pivotality-measuring IVs/aka the IVs I wasn't going to throw into a PCA.

In step 16, I scaled/standardized and ran the PCA with `splus`, `shapley`, `banzhaf`, and `miw_proportion`, and the first principal component of this PCA had an explained variance of 0.983, and was the linear combination `0.5013 * splus + 0.5034 * shapley + 0.5012 * banzhaf + 0.4941 * miw_proportion`.

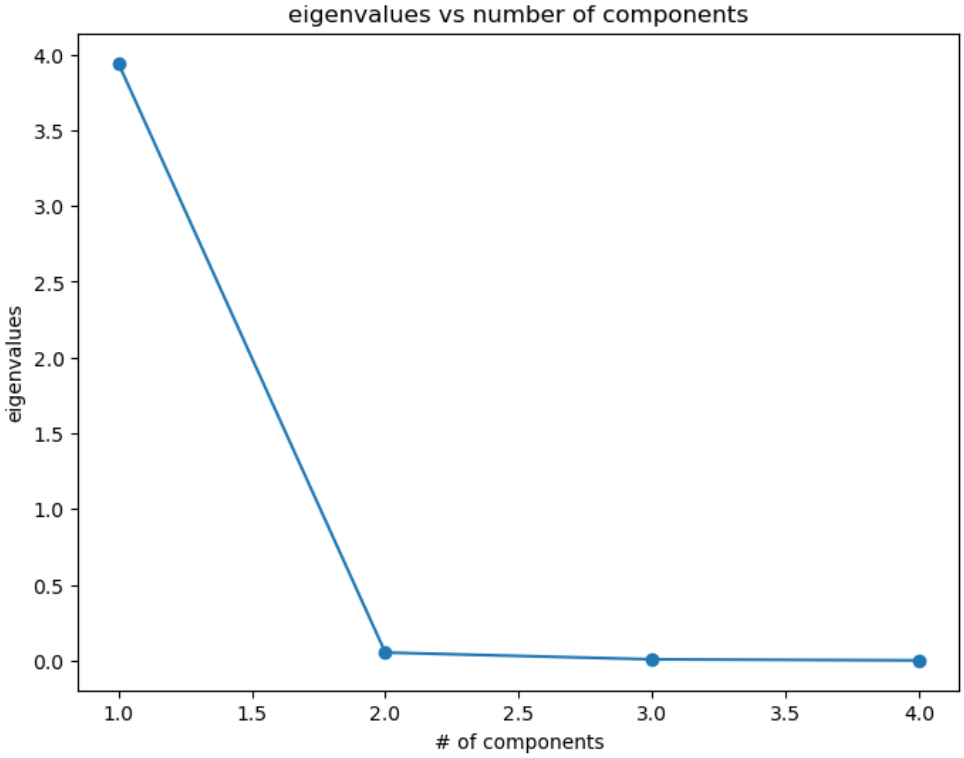
Note: much later in this data assignment, I went back and looked at the summary statistics for `PC0` (step 30). The min value of IV `PC0` measurements was -2.041597765620824, the max was 8.107807378257345, the mean was 0.09775958502160938, and the standard deviation was 2.059441900722237. The 25th, 50th, and 75th percentiles were -1.4280723888358509, -0.524538707487413, and 0.7415131195291416, respectively.

The second principal component had an explained variance of 0.0134, and so forth for the third and fourth principal components (they explained very little variance; much less than the second). The scree plot of explained variance vs. principal components is below. Cumulative explained variance, referred to as explained variance ratio in the y-axis of my scree plot, is terminology for what the ISLR textbook references as "proportion of variance explained by each of the four principal components" (James et al. 507).



I also plotted the eigenvalues of the PCA vs. the number of principal components; this plot is on the next page. The eigenvalue of the first principal component was 3.941. The second principal component had a principal component of 0.0538, and so forth for the third and fourth principal components (they had near-zero eigenvalues). Because the first principal component had a near-perfect proportion of variance explained, I was already learning on keeping on

the first principal component from the PCA, but I also did some reading on the Kaiser Rule, which suggests only retaining principal components with eigenvalues > 1 because eigenvalues > 1 indicate that the principal component explains more variance in the target variable compared to any other individual original IV. Because the eigenvalue of my first principal component was almost 4, the Kaiser Rule reinforced my decision to keep only the first principal component; in short, the Kaiser Rule helped simplify my selection of meaningful principal components.



After running PCA, I redefined my IVs and dataframe (added my principal component, which was named `PC0`, and removed the variables `splus`, `shapley`, `banzhaf`, and `miw_proportion` that were used to create latent variable `PC0`) in the last cell of step 16.

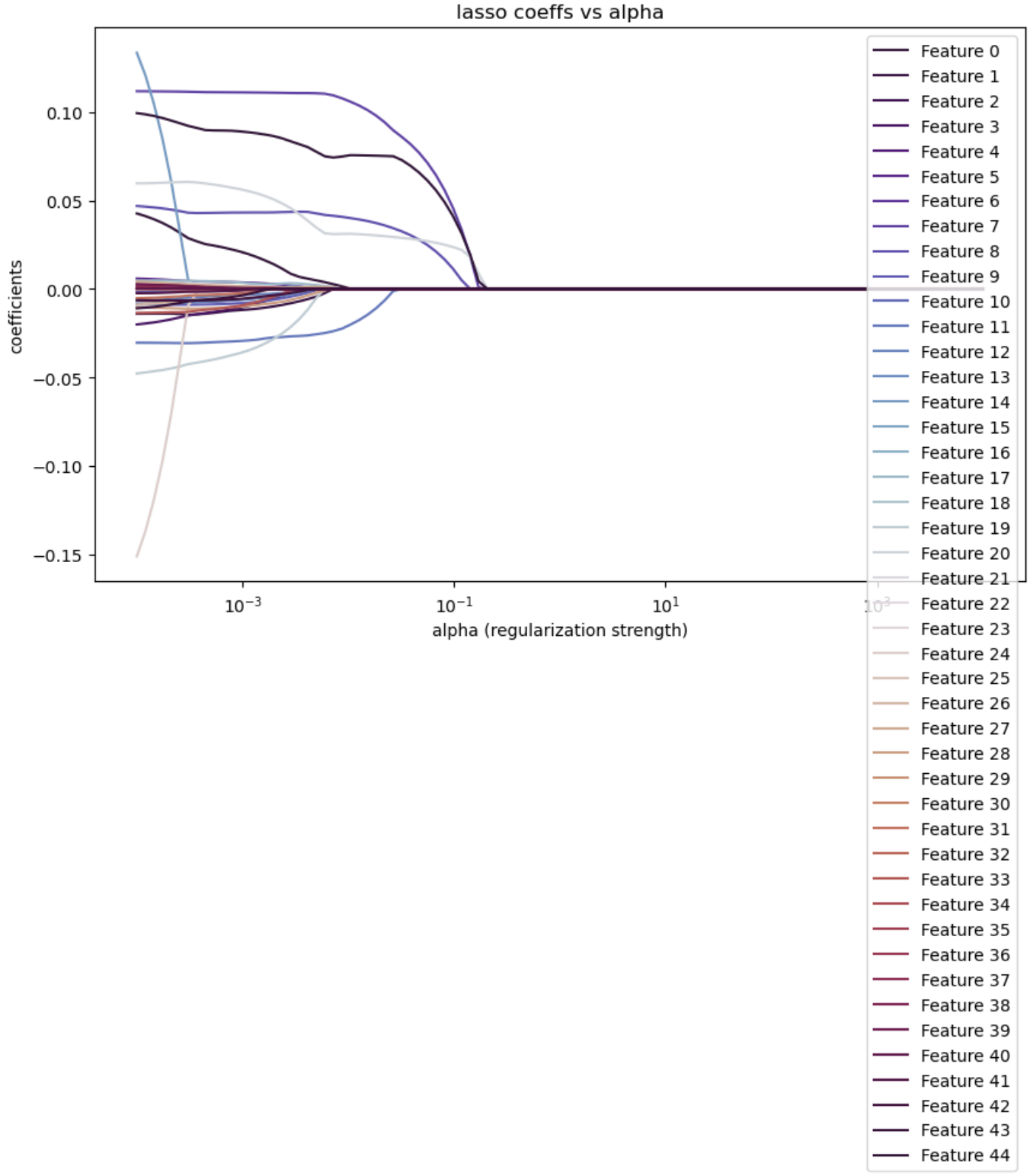
In step 17, I decided to run a k-fold with OLS to estimate model performance if I ran an OLS (not a polynomial regression) on this zero-inflated dataset. I had already established a 0.059 baseline MSE in a previous step, but I was curious about how bad it could get (because you're not supposed to run OLS on a zero-inflated DV). I used k=10, and the performance/error indicators for k-fold were average test MSE = 0.22 and average test R^2 = -3.60. The test R^2 of -3.60 was quite astonishing, and it underscored why OLS should not be used on zero-inflated data sets.

In my step 18 cell, I conducted a train/test split and double-checked that my PCA, appending of `PC0`, removal of the IVs used in the latent variable `PC0`, etc. worked. At this point, you had sent out in Slack that we shouldn't use `cabinet_seats` and `total_cabinet_size` as IVs, so I removed those. After removing those two IVs, instead of having 47 IVs total (46 IVs from the original data set, and `PC0`, the latent variable from my PCA), I now had 45 IVs (44 IVs from the original data set, and `PC0`, the latent variable from my PCA). In step 19, I decided to run a lasso in order to cut some more IVs (specifically the remaining IVs, which were not related to pivotality). The dataset was standardized/scaled before lasso, and I initialized a range of alpha values (parameters for regularization) through `np.logspace(-4, 4, 100)`. The plot of lasso coeffs vs. alpha (for all 46 IVs and `PC0`) is on the next page (labeled "Lasso Plot 1").

I didn't like my first lasso plot; unfortunately, the `matplotlib.cm` color range is limited, and it was difficult to actually identify the features using the legend. I couldn't tell which specific features were actually zero-ed out first by

lasso, and which ones weren't. Because the color gradient of Lasso Plot 1's legend was confusing, I ran another lasso where I dropped the first few IVs whose coefficients I knew would be shrinked to 0 first (from the first plot); Lasso Plot 2 was less confusing to look at.

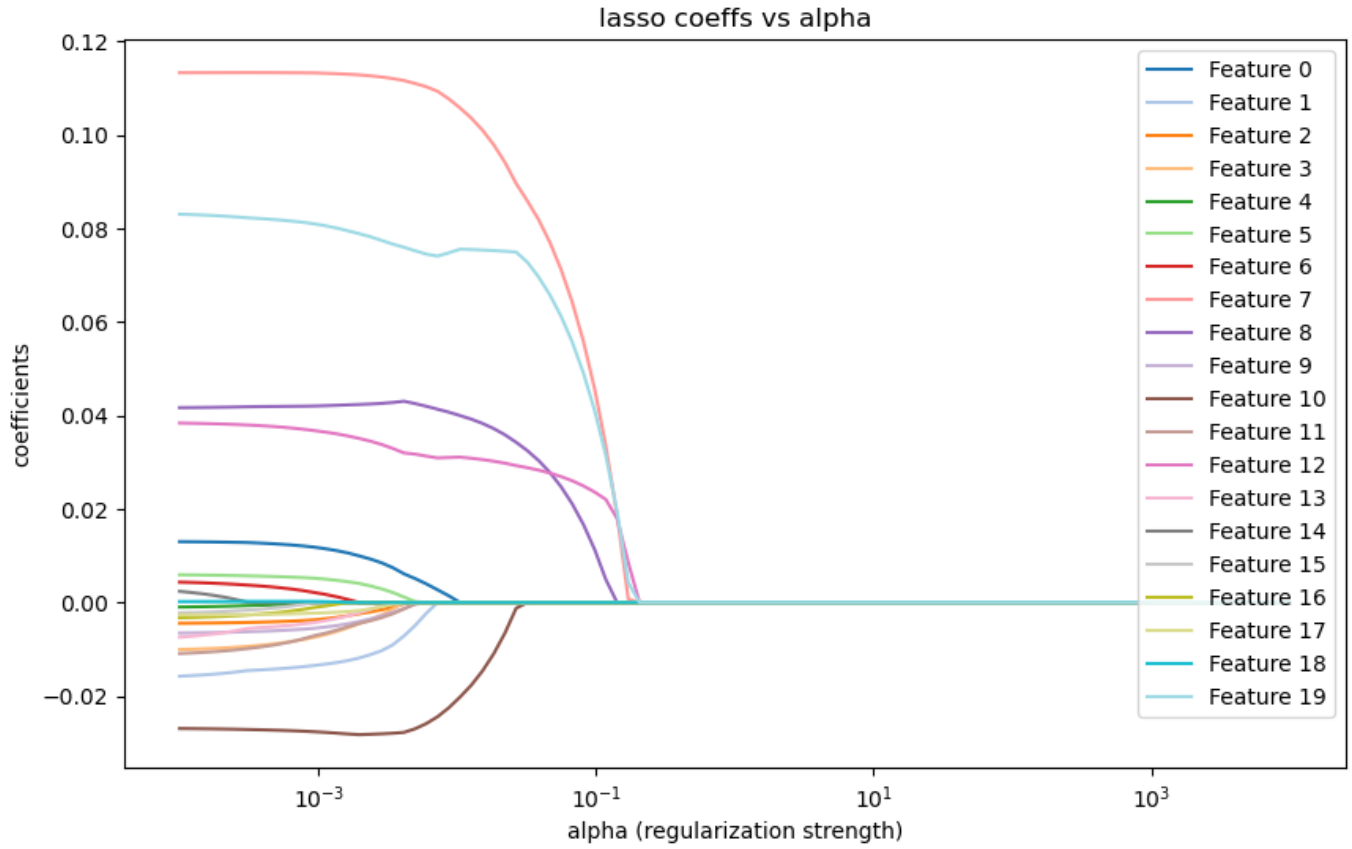
Lasso Plot 1:



List of features corresponding to the legend for Lasso Plot 1 (the # refers to the index in this list): `['seats',`

'sq_pm', 'election_year', 'miw_new', 'cabinet_id', 'party_id', 'caretaker', 'cabinet_party', 'prime_minister', 'left_rightx', 'party_count', 'cab_count', 'country_id', 'election_id', 'seats_share', 'post_election', 'enpp', 'mingov', 'bicameral', 'largest_parl', 'largest_cab', 'lag_largest_parl', 'lag_largest_cab', 'seats_total', 'seats_proportion', 'W', 'B', 'B_star', 'C', 'D', 'E', 'country_dummy1', 'country_dummy2', 'country_dummy3', 'country_dummy4', 'country_dummy5', 'country_dummy6', 'country_dummy7', 'country_dummy8', 'country_dummy9', 'country_dummy10', 'country_dummy11', 'country_dummy12', 'country_dummy13', 'PC0']

Lasso Plot 2:



List of features corresponding to the legend for Lasso Plot 2 (the # refers to the index in this list): ['seats', 'sq_pm', 'election_year', 'miw_new', 'cabinet_id', 'party_id', 'caretaker', 'cabinet_party', 'prime_minister', 'left_rightx', , 'cab_count', 'country_id', 'largest_cab', 'lag_largest_parl', 'lag_largest_cab', 'country_dummy10', 'country_dummy11', 'country_dummy12', 'country_dummy13', 'PC0']

*Notice how the second lasso plots fewer features; most of the IVs whose coefficients were shrunk to zero first were not plotted in the second lasso diagram.

Looking at Lasso Plot 2, there were four IVs that stood out – specifically the coefficients of the IVs that weren't reduced to zero until alpha was greater than 10^{-1}. These IVs were `prime_minister` (Feature 8), `cabinet_party` (Feature 7), `PC0` (Feature 19), and `largest_cab` (Feature 12). I decided to keep these four IVs, and dropped the rest.

In my first attempt at a poly (step 20), I standardized and used the four IVs `prime_minister, cabinet_party,`

`PC0`, and `largest_cab`. I also picked degree 2 (I wanted a simple regression model but with a degree > 1 because k-fold told me not to run OLS on this zero-inflated data set). Even though I only used four IVs, the resulting polynomial had 14 coefficients for the interaction and polynomial terms. This felt too confusing to me (I didn't have enough theory nor a strong enough understanding between all of the IVs to explain so much interaction). The train R^2 was 0.906789703979508, adjusted train R^2 of 0.902194830232019, test R^2 was 0.9027965943160292 and train MSE of 0.005354921612022739, test MSE of 0.004975774565847162. Even though the performance indicators were great (the train/test MSEs were comparable, my train/test R^2 were close together, my test MSE was well below my baseline MSE of 0.059 from step 9), I felt that a simpler model with fewer interaction terms between IVs existed.

My intuition behind adjusted train R^2 was that if all the correct IVs and perhaps a few noisy ones were included in my polynomial regression, if I took out the additional noisy/aka unnecessary variables, my RSS would increase slightly. This is because of the adjusted train R^2 equation: if you increase the number of IVs in your polynomial regression by adding noisy variables, you also increase RSS/(n-d), and consequently your adjusted R^2 decreases (not good!).

Adjusted R^2 formula from page 234 of ISLR textbook. Note that RSS * TSS = (1- train R^2)(N-1), and N = n = total sample size (number of observations), while d = p = number of IVs; these formulas for adjusted R^2 are interchangeable.

$$\text{Adjusted } R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}.$$

Another adjusted R^2 formula, the one used in my code:

$$Adjusted\ R^2 = 1 - \frac{(1 - R^2)(N-1)}{N-p-1}$$

Where

R^2 Sample R-Squared

N Total Sample Size

p Number of independent variable

Source: https://medium.com/analytics-vidhya/adjusted-r-squared-formula-explanation-1ce033e25699

After running my first poly attempt, I wanted to increase my adjusted R^2 and decrease the number of poly/interaction terms I had, so I decided to do more feature selection and find a simpler model.

I went back and looked at my earlier VIF table and correlation matrix (steps 10-12). `prime_minister` had a VIF of 5.271343e+00 in my earlier VIF table, while `largest_cab` had a VIF of 6.248973e+00, while `cabinet_party` had a VIF of 4.032470e+00. In short, both `prime_minister` and `cabinet_party` had VIF values that were higher than the ISLR's threshold of 5 as an indicator for multicollinearity. I did not have a VIF measurement for `PC0`, because I ran PCA in step 16/my latent variable did not exist when I ran the VIF in step 10. Also, `prime_minister` was correlated with `banzhaf` at 0.67 and `shapely` at 0.68 and `miw_proportion` at 0.63 (three of the four variables used to create my latent variable). Also, `largest_cab` was correlated with `banzhaf` at 0.68 and `shapley` at 0.69 and `splus` at 0.71 and `miw_proportion` at 0.65 (all four variables used to create my latent variable `PC0`). Because it was highly probable that there was multicollinearity between `PC0` and `prime_minister`

and largest_cab, I decided to drop prime_minister and largest_cab. I did not drop cabinet_party because it had a VIF value of 4.032470e+00, which was under ISLR's threshold of 5, and it had no correlation with any of the pivotality measuring IVs that were used to create my latent variable PC0.

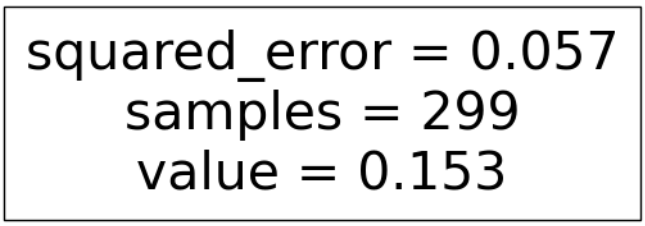
I now had two remaining IVs: PC0 and cabinet_party. I ran my second polynomial regression (step 24), and this poly only had 5 terms (down from 14 in my first poly). For performance indicators, my train R^2 went up to 0.9157664387359933, adjusted train R^2 went up to 0.9143290059499181, test R^2 went up to 0.9078263599442475, and train MSE went down to 0.0048392091531509135 and test MSE went down to 0.004718304370137877 ("down to" and "up to" phrasing here is in comparison to my first poly). Because my adjusted train R^2 went up between the two polys and after I dropped prime_minister and largest_cab, I believe that prime_minister and largest_cab were both noisy/unnecessary variables. However, this polynomial is still lacking in interpretability, especially since the interaction term exists and would require theory/more knowledge of coalition governments and parties that I don't have. I also don't fully understand why some of the coefficients have the positive/negative signs that they have, and my interpretation of or story behind the interaction terms is slightly crazy/non-existent.

Final polynomial equation: cabinet_proportion = −201926564546.0759 + 8.50402461e-02 * PC0 + -6.70903266e+10 * cabinet_party + -3.17728009e-03 * $PC0^2$ + 9.79062350e-02 * PC0 * cabinet_party + 2.01926565e+11 * $cabinet\_party^2$

Interpretation of final polynomial equation: according to ISLR, interaction terms are the extension of a linear model, and I'm not sure if it's possible to take them out of a polynomial regression in sklearn. Interaction terms show a synergistic relationship between IVs; in other words, the effect of the PC0 on cabinet_proportion depends on the value cabinet_party (a change in the first IV in an interaction term leads to a change in the association between the other IV in the interaction term and the DV). cabinet_party is a binary variable for whether a party is in the winning coalition, and PC0 is a latent variable measuring the amount of pivotality a party has. The presence of the interaction term suggests that if a party becomes part of the winning coalition (i.e., their score on cabinet_party goes from 0 to 1), then their pivotality somehow increases cabinet_proportion. On the other hand, the interaction term also means that if a party increases their pivotality PC0 to a certain threshold, they suddenly join the winning coalition (i.e., their score on cabinet_party goes from 1 to 0) and their cabinet_proportion increases. This makes some sense, but again, I lack theory and a background of coalition governments, but I'm doubtful that any party could suddenly switch on the binary from being part of a winning coalition to not or vice versa because of a slight change in pivotality; I'm not sure that those scenarios would be realistic, and as a result, I'm not entirely confident in this polynomial. Also, I don't fully understand why $PC0^2$ has a negative coefficient, while PC0 has a positive one, and why $cabinet\_party^2$ has a positive coefficient, while cabinet_party has a negative one. This makes the story behind the polynomial slightly crazy.

Higher values of PC0 indicate a higher value of a linear combination of the pivotality measures, and if a party has a high PC0 measurement, it implies the party had high pivotality measured among splus, shapley, banzhaf, and miw_proportion.

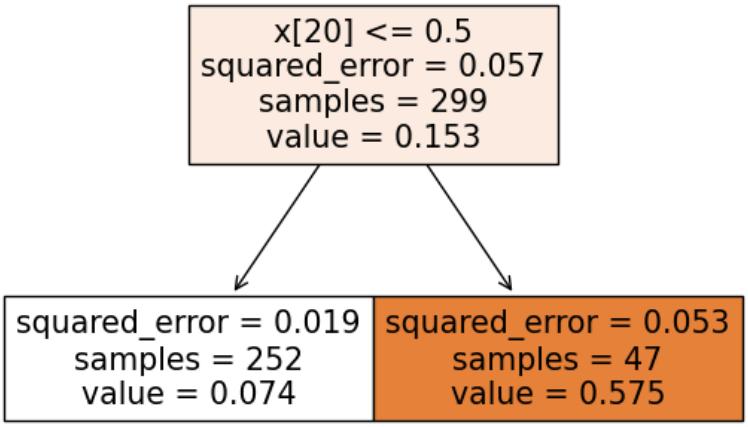
I created several decision trees. In step 25, I decided to run a decision tree with the two IVs I used in my final polynomial equation (I knew at the time that this wasn't a bright idea, because two IVs is not enough for a tree to split on/going from 45 IVs to 2 IVs might be too much feature selecting for a tree). My tree looked like this:



I was confused, because it looked like there were no splits, and I had a single root node. The performance indicators were a train R^2 of 1.0, a test R^2 of 0.848391306082719, train MSE of 0.0, and test MSE of

0.0077607433386391 44. The alpha value was found through cost complexity analysis in the `ccp_alpha=alpha` line in the previous cell/I searched for alpha values in that cell for the best estimated test MSE. I didn't bother comparing this decision tree's performance indicators to my final poly or baseline, because this tree was completely uninterpretable.

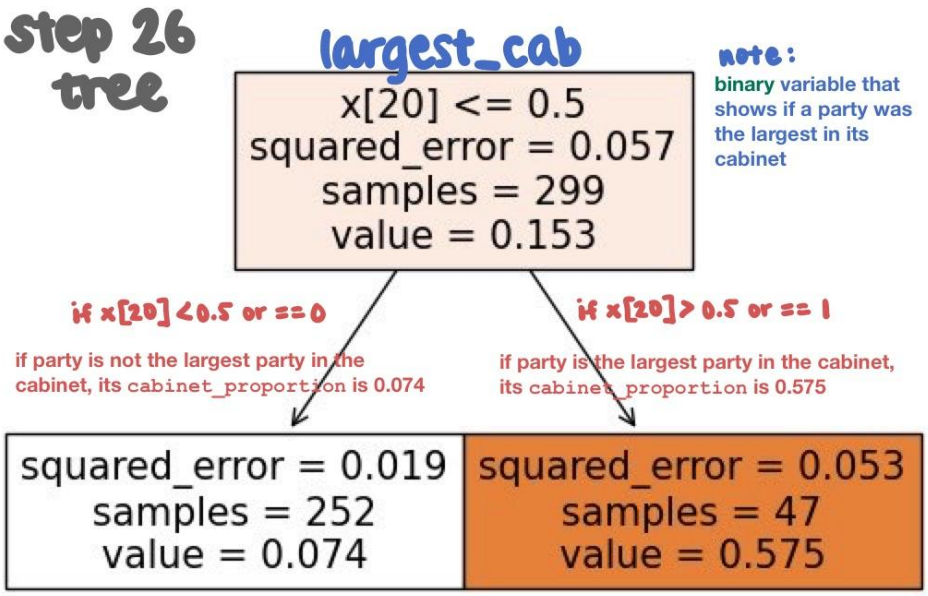
I ran another decision tree (step 26), but this one was fed in all possible 45 IVs from `ind_vars` (`cabinet_seats` and `total_cabinet_size` taken out from the 47 IVs in the original `updated_data` frame; of the 45, 44 IVs were from original data set, and the last IV in `ind_vars` is `PC0`). In other words, I did little feature selection for this decision tree. The visualization for this tree is below.



This tree had a single split; **`x[20]` refers to `largest_cab`**. I thought this was interesting, because I had taken largest_cab out of my poly since it had a VIF > 5, and had believed that it was a highly multicollinear variable. This tree's choice of split demonstrated how decision trees can't identify and prevent multicollinearity, especially when one doesn't feature-select before throwing variables into a decision tree. See the annotated version of the tree for my interpretation.

This decision tree had a train R^2 of 1.0, a test R^2 of 0.8855905896712042, train MSE of 6.183587781310189e-35, and test MSE of 0.005856537947429855. The alpha value was found through cost complexity analysis in the `ccp_alpha=alpha` line in the cell of the previous tree. Performance indicators for this tree were fine (the train/test MSEs were comparable, my train/test R^2 were good enough, my test MSE was well below my baseline MSE of 0.059 from step 9), but the tree overfit to my training set (something may have gone wrong in my alpha value search), and I didn't like how its single split was on an IV that I had identified as multicollinear in previous steps. With my limited knowledge of coalition governments, this tree made some sense: if a party was the largest in its cabinet, its awarded proportion of cabinet ministries must be high. However, this tree didn't take pivotality into account (it didn't incorporate my latent variable `PC0`), and again, I still didn't like how its single split was on a multicollinear IV.

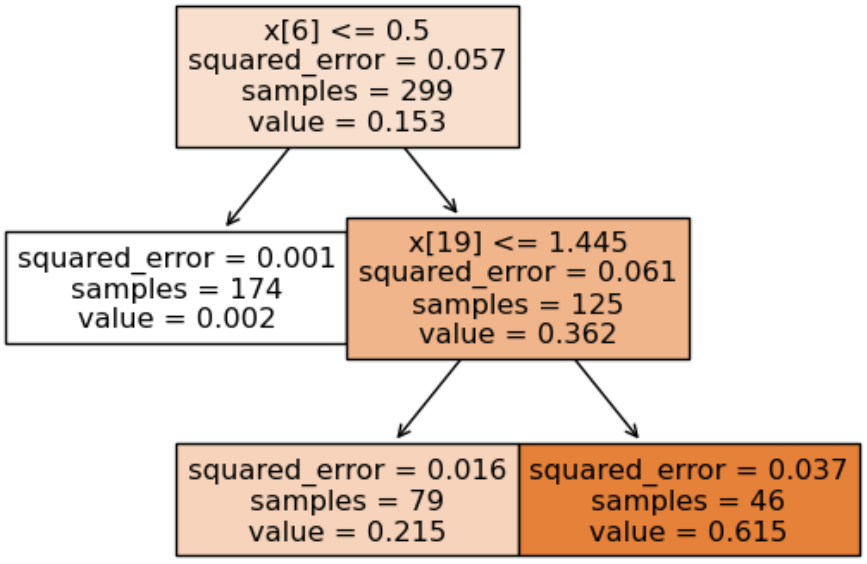
Annotated version of this tree:



In step 27, I decided to run lasso again, but this time for the sole purpose of conducting feature-selection for a decision tree. At this point, I had realized that I needed a happy medium of feature selection—between the **extremes** of trying to get a handful of IVs (for polys) **and** throwing all possible IVs (risking single-split trees on multicollinear variables)—for trees. I picked an alpha value/regularization parameter of 0.001. In hindsight, perhaps I should've used CV to search for the optimal alpha value to use, but I was running out of time. The list of IVs (from the 45 total, as stated earlier) that still had non-zero coefficients after running a lasso with an alpha of 0.001 was `['seats', 'sq_pm', 'election_year', 'miw_new', 'cabinet_id', 'party_id', 'cabinet_party', 'prime_minister', 'left_rightx', 'party_count', 'cab_count', 'country_id', 'election_id', 'enpp', 'mingov', 'largest_parl', 'largest_cab', 'seats_total', 'W', 'C', 'D', 'PC0']`. I initialized this list as `variables_nonzeroed`.

At this point, I had a feeling that a new tree (fed with the IVs from `variables_nonzeroed`) was going to split on `largest_cab` again. I was correct; the tree I ran (with the IVs from `variables_nonzeroed` fed in) split on `largest_cab` (step 28). I did not bother checking the performance indicators, and removed `largest_cab` and `prime_minister` (another IV I had identified as multicollinear while conducting feature selection for my second poly) from `variables_nonzeroed`, and then ran a new tree with the otherwise same list of IVs (step 29).

The tree I ran in step 29 was forced not to split on `largest_cab` or `prime_minister`, because I feature-selected those two IVs out of the IVs it could choose from, but it could choose from all of the other IVs in the list `variables_nonzeroed`. For this tree, I set the `ccp_alpha` value to 0.01 (for no particular reason other than I wanted to try a `ccp_alpha` that wasn't forcing my trees to be single-splits). In hindsight, I should've run specifically k-fold cross validation to ensure that the value of alpha I was choosing did not create a tree that would overfit to my training set. Regardless, the visualization for this tree is on the left, and the annotated version is on the next page.



`x[6]` refers to `cabinet_party`

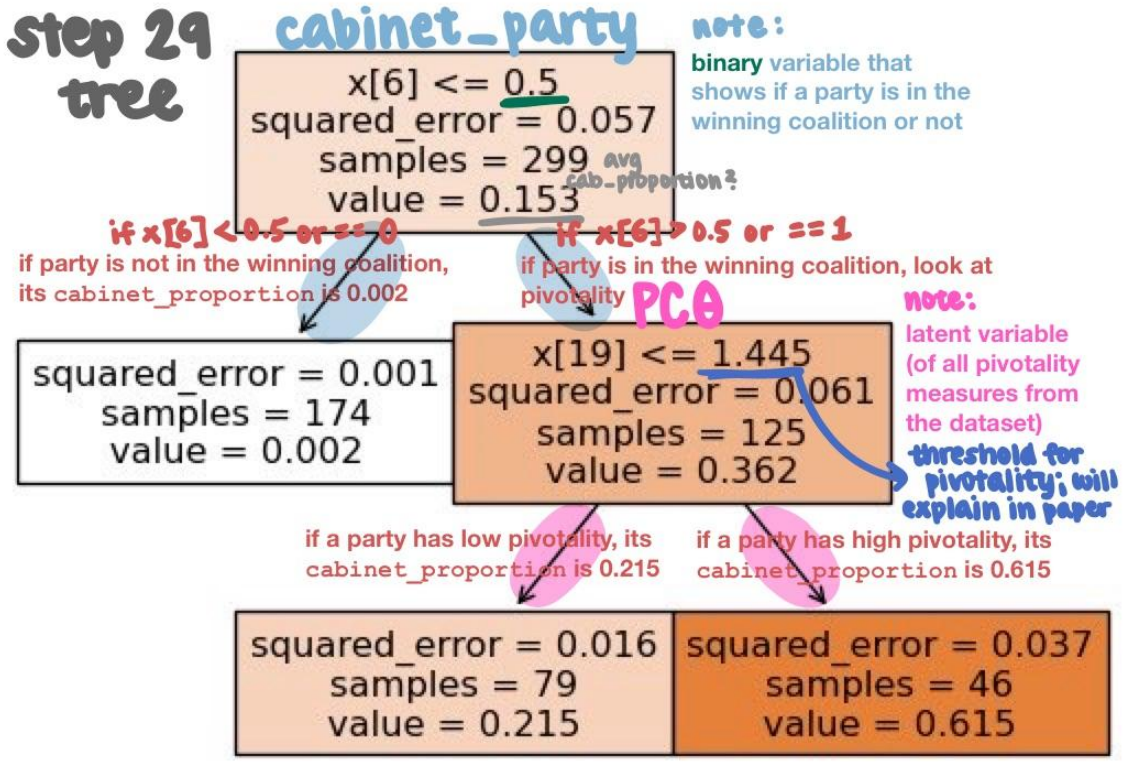
`x[19]` refers to `PC0`

The model performance for this tree was a train R^2 of 1.0, test R^2 of 0.8895608931555545, train MSE of 0.0, and test MSE of 0.005653300880198345. I thought that the train/test MSEs were comparable, and the test MSE was still well below my baseline MSE of 0.059 from step 9. I noticed that this tree seemed to overfit on my train set, which is not ideal. Even though this tree had a lower test R^2 compared to my final polynomial regression (0.8895608931555545 vs. 0.9078263599442475) and a higher test MSE compared to my final polynomial regression (0.005653300880198345 vs. 0.004718304370137877), I think this tree (with its two splits) is more interpretable than the polynomial regression (which has 5 total poly and interaction terms). I'd also like to note that the inequality data assignment taught me that having an R-squared lower than 0.9 is not the end of the world, especially in the social sciences, so I wouldn't consider this decision tree's performance to be poor at all.

If I had to pick the better model between the two, the sacrifice of model predictability (lower test R^2 and higher test MSE) is worth it for the tree, because the tree is significantly more interpretable than my final polynomial regression, even though both used the same two IVs. I was not confident in my interpretation of the interaction term in the polynomial regression, and didn't fully understand the signs of the coefficients for the other terms.

In short, I liked this tree because the splits made more sense to me than the interaction and polynomial terms in my final polynomial regression equation, and this tree used the same IVs that were in my final poly. I decided that this tree was my final decision tree, and that this final decision tree was my best model/story about this data set.

Annotated version of this tree:



Interpretation of final tree (step 29): if a party is not in the winning coalition, then its proportion of cabinet ministries awarded is effectively 0; the tree sets `cabinet_proportion` to 0.002. If a party is in the winning coalition, then the tree looks at the party's pivotality. `cabinet_party` is the most important factor in determining `cabinet_proportion`. If a party's pivotality is less than or equal to a threshold of 1.445 (see next paragraph for more explanation of threshold), then the party's awarded proportion of cabinet ministries is 0.215. If a party's pivotality is greater than a threshold of 1.445, its awarded proportion of cabinet ministries is much greater; the tree sets `cabinet_proportion` to 0.614.

Explanation of `PC0`/threshold of 1.445 in the context of final decision tree: note that PC0 is a latent variable; it is composed of the measures of pivotality in the data set (`splus`, `shapley`, `banzhaf`, and `miw_proportion`) and is the linear combination `0.5013 * splus + 0.5034 * shapley + 0.5012 * banzhaf + 0.4941 * miw_proportion`. Because `splus` ranged from 0.003051758 to 0.78536028, while `shapley` ranged from 0.0 to 0.71428573, and `banzhaf` ranged from 0.0 to 0.83783782, and `miw_proportion` ranged from 0.0 to 0.45454547, they each had a slightly different scaling. Even though I scaled/standardized before running PCA to get the linear combination equation for the latent variable `PC0` (see first sentence of this paragraph), it's difficult to explain what exactly makes the pivotality threshold of 1.445 in the decision tree so significant in terms of the IVs that make up the latent variable (`splus`, `shapley`, `banzhaf`, and `miw_proportion`). In general, higher values of PC0 indicate a higher value of a linear combination of the pivotality measures, and if a party has a high PC0 measurement, it implies the party had high pivotality measured among `splus`, `shapley`, `banzhaf`, and `miw_proportion`. I realize that my interpretation of the threshold is uninsightful, but I still feel better about it than I do about the interaction term of the polynomial.

In step 31, I ran `xgboost`. The `xgboost`'s train MSE was 4.0527830246743727e-07, test MSE = 0.001159053603868567, train R^2 = 0.9999928892894258, test R^2 = 0.9776479323589424. Because `xgboost` seems to be slightly overfit on its training data (test R^2 is slightly smaller than train R^2, and test MSE is greater than train MSE), I felt marginally better about the fact that my final decision tree was also overfit on training data. `xgboost's` model performance indicates that a model more accurate or with better predictability than my final polynomial regression or final decision tree exists out there; however, that model would likely going to be difficult for me and my lack of theory to interpret (considering the fact that I already struggled to interpret the coefficient signs of my polynomial, and was doubtful about its interaction term).

In conclusion, my final decision tree was more interpretable than my final polynomial regression. Although the tree did slightly worse than the polynomial on predictability (I compared the performance indicators test MSEs and test R^2s between the two models) and was consistently overfit on my training data set, I find it acceptable to sacrifice predictability for interpretability. There is a chance that the decision tree oversimplifies the relationship between `cabinet_proportion`, `PC0` (and by extension, `splus`, `shapley`, `banzhaf`, and `miw_proportion`), and `cabinet_party`; however, its advantages of interpretability and graphical representation outweigh its cons.

I also attribute my preference for model interpretability to my lack of theory. As someone who doesn't fully understand the theory behind this data set and all of the possible theoretical relationships between the IVs, the tree provides a simple, understandable model for what proportion of cabinet ministries a party is awarded. I prefer that over a black box model (`xgboost`) or one with an interaction term or coefficients that I'm not convinced at all by (poly). If I ever go back to this data set, I plan on 1) exploring whether or not it's possible to get rid of interaction terms in a polynomial regression in sklearn and 2) implementing a k-fold cross validation to search for an alpha value to create a non-overfit decision tree.