***Report – HW4***

Stable Coalition:

For built stable coalition, we look on the problem from 2 different side:

1. Clustering – split all examples to two different groups using a distance function using continuous values, when in each group has similar examples. the big group is the coalition and the smaller group is the opposition.
2. Generative modeling – we will use the probability function properties to compare the different parties to find their similarities and differences. for example, for gaussian naïve base, we can get from the model the expectation and variance and use them to compare between the different parties and build two different groups, the big group is the coalition and the smaller group is the opposition.

Stable Coalition using clustering:

We use 2 models that we learn in the class for clustering model K-means and Gaussian mixture. first, we found the hyper parameters using k fold cross validation, we define evaluate function to evaluate the best hyper parameters. The evaluate function is to find a group that contains as many voters from a specific party and does not split voters into several different groups. we set a certain threshold value from which we decide that a group belongs to a specific group, for example 45% for k=3 clusters.

The process the we made to reach stable coalition is:

1. we train the train set using k-means for different k clusters and for different threshold.
2. we check accuracy on the validation set and tuning hyper parameters (k and threshold) accordingly.
3. Finally, we check accuracy on the test set.

Accuracy:

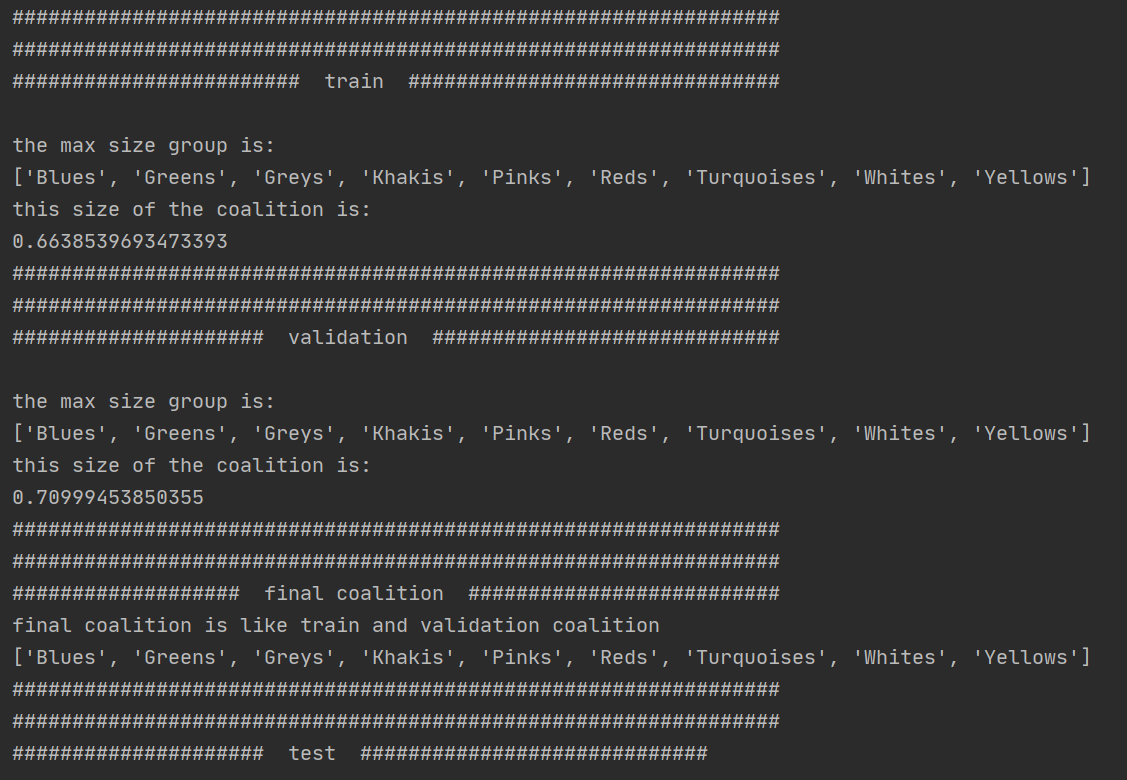
How we check accuracy in the test and validation set?

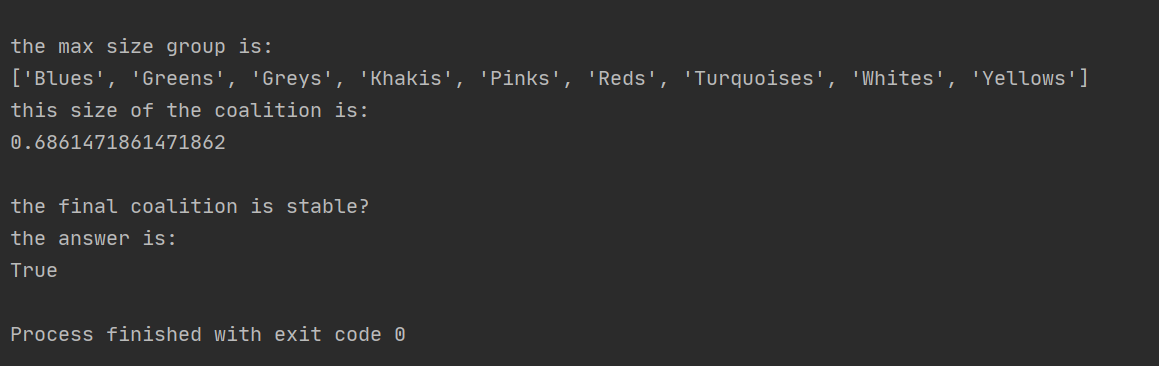
We check if we get equal coalition in all sets, that is:

Train coalition == validation coalition &&

Test coalition == validation coalition == train coalition

Finally, we get these results:





We can see from these results that we get stable coalition for k=3 and threshold=0.45,

The final coalition consists of 9 parties and they:



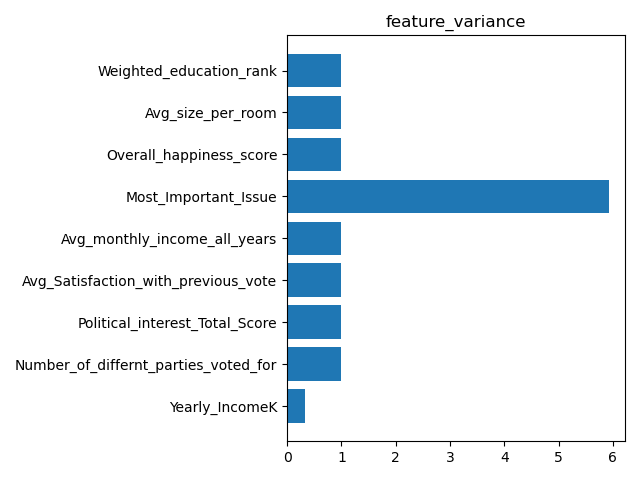
Size: The sizes of the coalitions in different sets is:

Train: 66.3%, validation: 71%, test: 68.6%

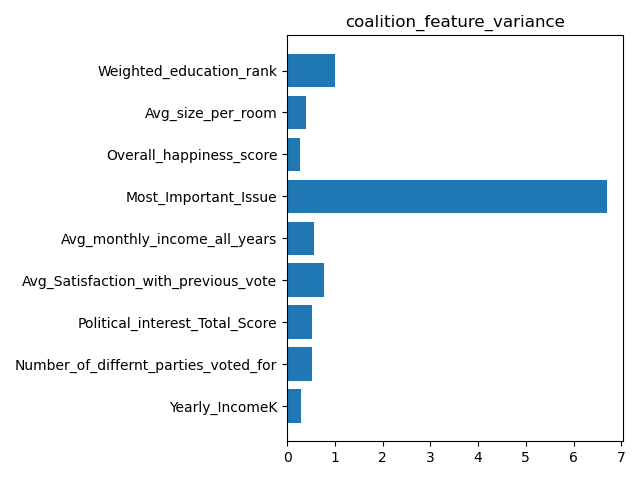
We can see that the sizes are about the same size.

Homogeneous: The homogeneous of the coalition is determined by threshold, the bigger the threshold the bigger the homogeneous. after a lot of experiments, for different k and different threshold, we saw that we get stable coalition for k=3 and threshold=0.45, and if we take threshold that bigger from k=0.45, we did not accept stable accuracy.

Before choosing coalition we get variance like this:



And after choosing coalition we get:



We can see from the graphs that the variance is smaller, hence the coalition is more homogeneous.

***Stable Coalition using Generative Model:***

Our main idea here is to train a generative model like naïve gaussian bayes.   
The model will give us the party which has the highest probability of being elected.  
We will first add this party to the coalition, and then try to constitute a coalition around this party.

For each of the remaining party, we evaluate the features’ variances of the current coalition if the party is added to it. If globally the variance is reduced, we add it to the group. Else we discard it.

For instance:

**[graph before]**

Here are the variances at a certain point of our algorithm.

Then we evaluate the new variances if we add a party X.

**[graph after]**

We see that globally the variances are reduced, so we will keep this party.

This way, we get a homogenous coalition.

Eventually, we check that we get more than 51% of votes with the selected coalition, using the test set.

Let us now compare the variances of the coalition previously picked automatically by the clustering model, and the variances of the new coalition picked manually, using the generative model:

**[coalition variance cluster]**

**[coalition variance gen]**

It seems that the manual job gives us a better homogeneity.

***The leading features of each party:***

To identify the leading features of each party, we will have a look at the variances of the voters’ features for each party. Each feature having a low variance indicates us that most of the voters for a given party have more or less the same value a feature. This means that it’s a strong characteristic of the party.  
Therefore, we will print the variances for each party to determinate what are the leading features.

**[Graph for one]**

**[Interpretation for one]**

Similarly, we do the exact same operation for each party, giving us the following results:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

***Manipulating the elections***

After doing many manipulations, we found out that modifying the Yearly\_IncomeK and the Number\_of\_differnt\_parties\_voted\_for has no impact on the elections’ winner.  
However, every other modifications has an impact:

|  |  |  |
| --- | --- | --- |
| Feature | Operation | Winner |
| Political\_interest\_Total\_Score | +24 | Browns |
| Avg\_Satisfaction\_with\_previous\_vote | +2 | Browns |
| Avg\_monthly\_income\_all\_years | +0.5 | Browns |
| Most\_Important\_Issue | +4 | Browns |
| Overall\_happiness\_score | +3 | Browns |
| Avg\_size\_per\_room | +3 | Purples |
| Weighted\_education\_rank | +3 | Browns |

(The operations were made on the prepared test data set)

***Strengthening the coalition***

[Take the current coalition and modify one by one columns to see if more vote eventually or not]