***Report – HW4***

***Stable coalition:***

There is two ways to build a stable coalition:

1. Clustering – Split the data in two different groups using a distance function with continuous values, when in each group is homogenous. The biggest group is the coalition and the smallest one is the opposition.
2. Generative modeling – We use a generative model to predict the winner of the elections, among other information like the variance of the features for a given group of party to generate the best coalition possible. The coalition will be constituted around the most probable winner.

***Stable coalition using clustering:***

We are going to use the clustering model we studied in class, K-Means.  
By using cross validation, we approximated the best hyper parameters for the model.

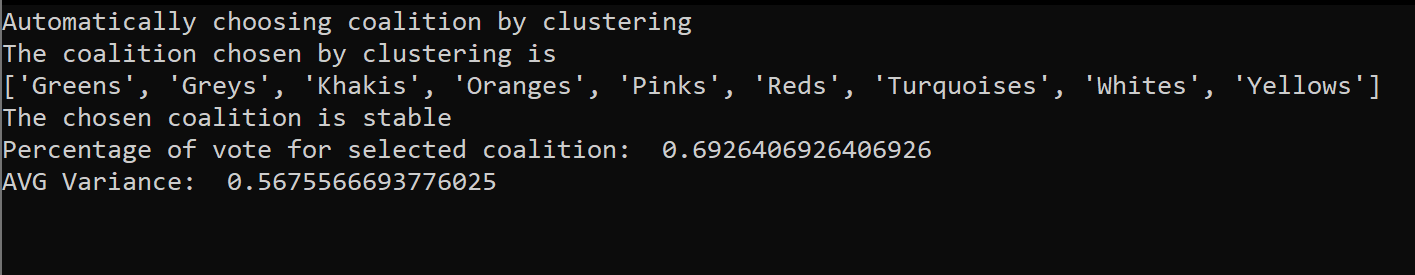
Then, using clustering we will group the similar voters and get their parties, in order to form a homogenous coalition.

We set a certain threshold value from which we decide that a voter belongs to a specific group, for example 45% for k=3 clusters.

We used the following process to form a stable coalition:

1. Training the K-Means models with the train set with different k values and threshold.
2. Checking the accuracy on the validation set and tuning hyper parameters (k and threshold) accordingly.
3. Checking the accuracy on the test set.
4. Then, form a coalition by getting parties belonging to the biggest cluster.
5. To get a coalition which is different than the opposition, we check that a given party is not overrepresented in the other clusters.

Finally, we get these results:



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

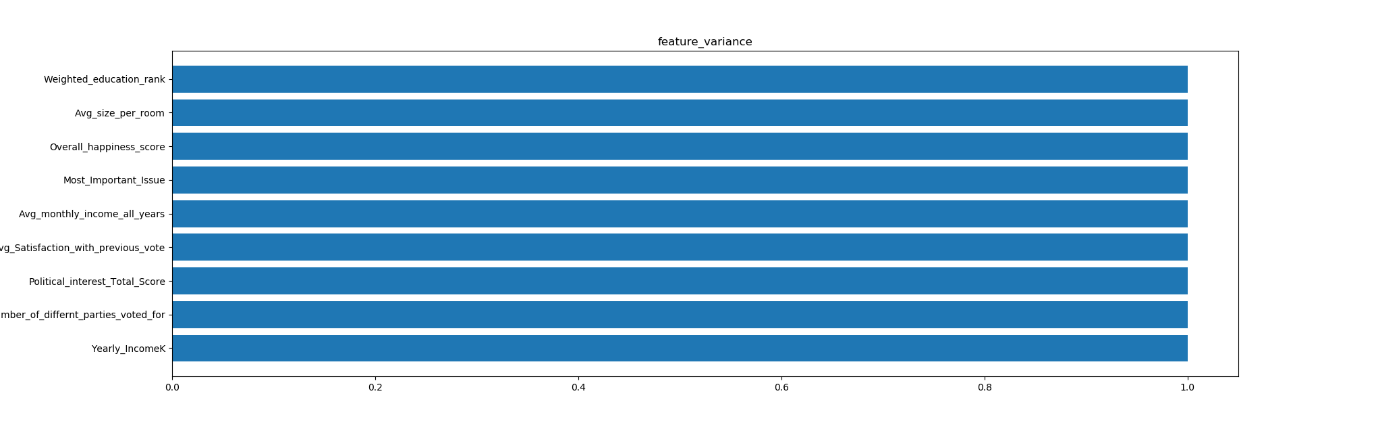
The final coalition consists of 9 parties, which are:



We get 69.26% of the votes in the test set with this coalition.

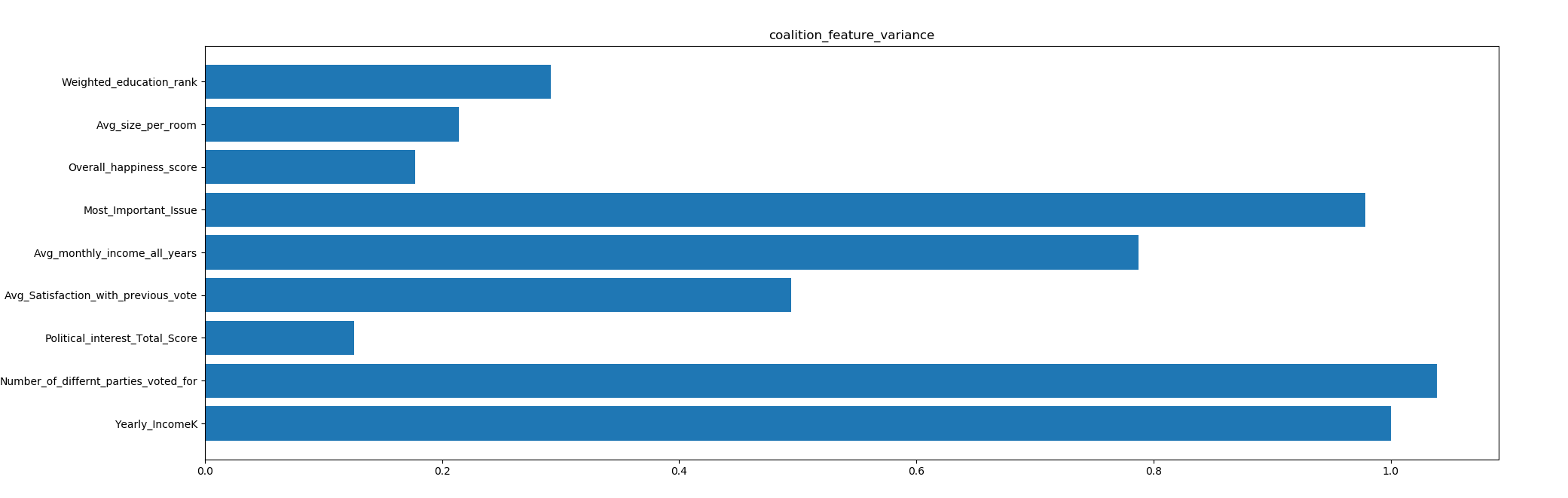
Homogeneous: The homogeneity of the coalition is determined by threshold. A bigger threshold gives a better the homogeneity. After lots of experiments, for different k and different threshold, we determined that we get stable coalition for k=3 and threshold=0.45, and if we increase the threshold more than k=0.45, the results are not good enough.

Before choosing coalition we got variance like this:



(Every feature has a variance of 1)

And after choosing coalition we get:



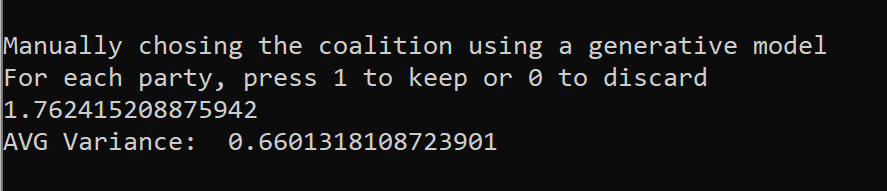
We can see from the graphs that the variance is smaller, hence the coalition is more homogeneous. This gives us an average variance of 0.5676

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

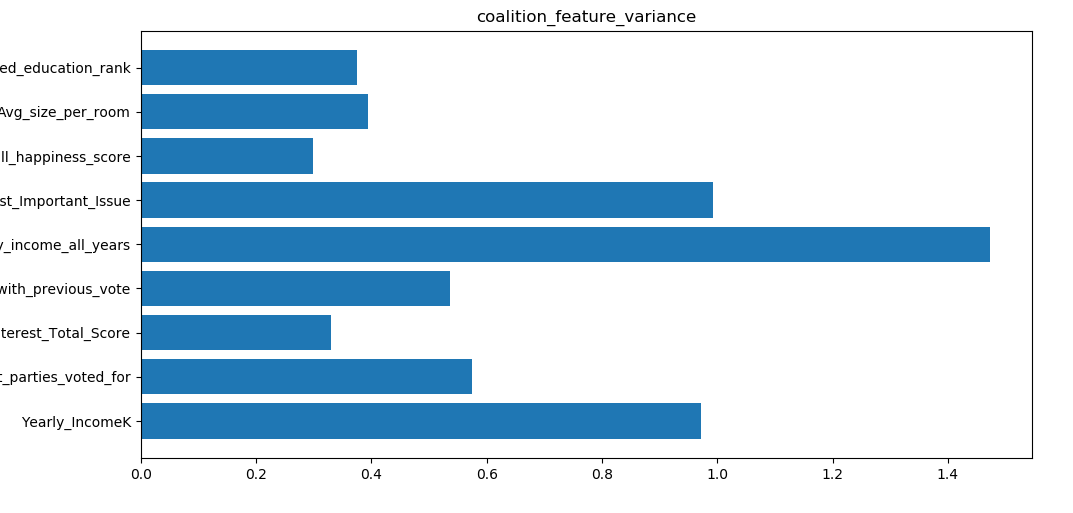
Our main idea here is to train a generative model like naïve gaussian Bayes.   
The model will give us the party which has the highest probability of being elected.  
We will first add this party to the coalition, and then try to constitute a coalition around this party.

For each of the remaining party, we evaluate the features’ variances of the current coalition if the party is added to it. Also, we determine the distance score between the coalition and the opposition, by computing the Euclidian distance of the average variance vectors. If for a given party, the variance is reduced and the distance is increased, we add it to the group. Else we discard it.

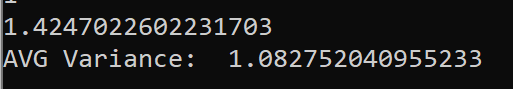
For instance, at one step of our algorithm we can get this output:

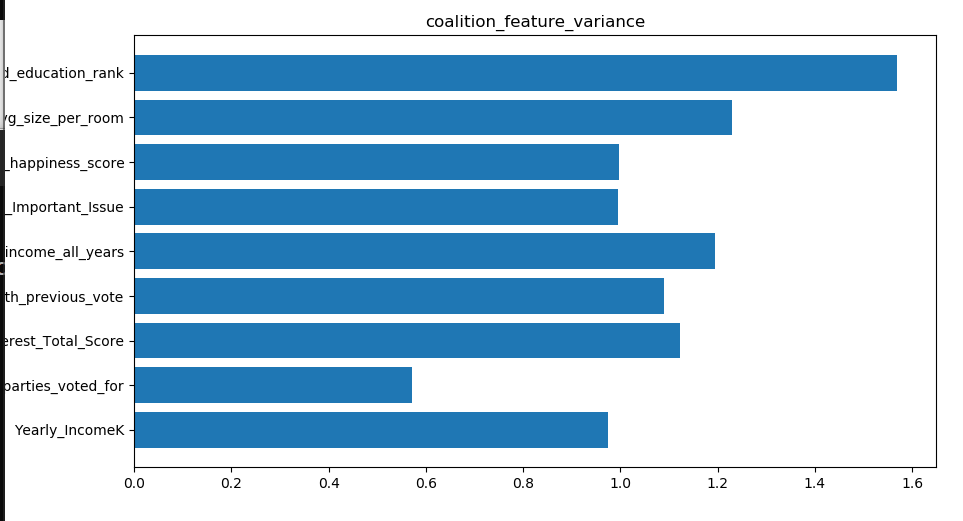


(The first number is the Euclidian distance score between the coalition and the opposition)



And after considering a new party to the coalition, we get those results:

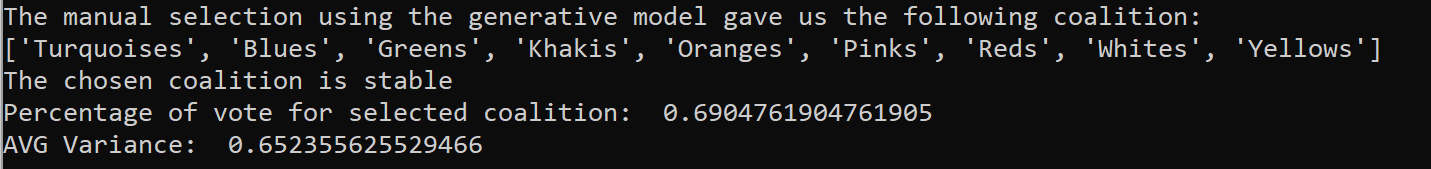


We see that globally the variances are increased, and the difference between the coalition and the opposition decreased, so we will not keep the party in the coalition.

If we got instead an increased distance and a decreased average variance, we would have kept it.

This way, we get a homogenous coalition, and also much different from the opposition.

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

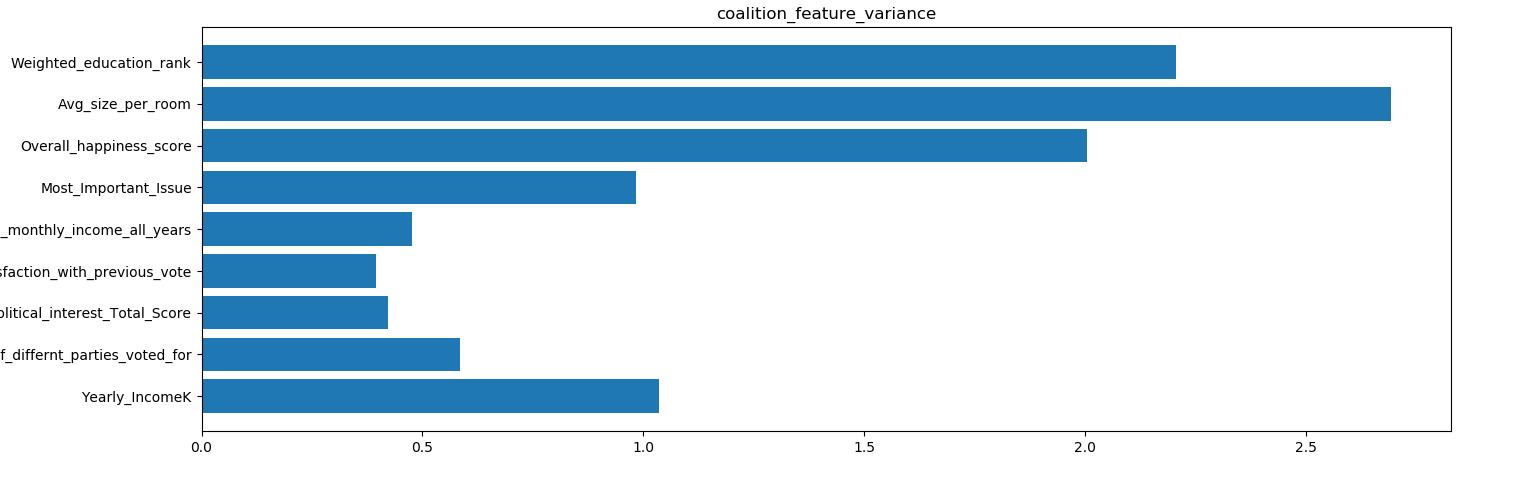


We can see that it gave us a coalition very similar to the one found with the clustering method.

***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 is a strong characteristic of the party.  
Therefore, we will print the variances for each party to determinate what are the leading features.

For instance, we are the variances for the Blue party:



We can see that the monthly income, the satisfaction with the previous vote and the political interest have a low variance, below 0.5. Therefore, we can conclude that they are the leading features of the party.  
Similarly, we do the exact same operation for each party, giving us the following results:

|  |  |
| --- | --- |
| Blues | ['Political\_interest\_Total\_Score' 'Avg\_Satisfaction\_with\_previous\_vote'  'Avg\_monthly\_income\_all\_years'] |
| Browns | ['Avg\_Satisfaction\_with\_previous\_vote'] |
| Greens | ['Political\_interest\_Total\_Score' 'Avg\_Satisfaction\_with\_previous\_vote'  'Overall\_happiness\_score' 'Avg\_size\_per\_room' 'Weighted\_education\_rank'] |
| Greys | ['Number\_of\_differnt\_parties\_voted\_for' 'Political\_interest\_Total\_Score'  'Avg\_Satisfaction\_with\_previous\_vote' 'Most\_Important\_Issue'  'Overall\_happiness\_score' 'Avg\_size\_per\_room' 'Weighted\_education\_rank'] |
| Khakis | ['Number\_of\_differnt\_parties\_voted\_for' 'Political\_interest\_Total\_Score'  'Avg\_Satisfaction\_with\_previous\_vote' 'Overall\_happiness\_score'  'Avg\_size\_per\_room' 'Weighted\_education\_rank'] |
| Oranges | ['Number\_of\_differnt\_parties\_voted\_for' 'Political\_interest\_Total\_Score'  'Avg\_Satisfaction\_with\_previous\_vote' 'Most\_Important\_Issue'  'Avg\_size\_per\_room' 'Weighted\_education\_rank'] |
| Pinks | ['Number\_of\_differnt\_parties\_voted\_for' 'Political\_interest\_Total\_Score'  'Avg\_Satisfaction\_with\_previous\_vote' 'Overall\_happiness\_score'] |
| Purples | ['Avg\_Satisfaction\_with\_previous\_vote'] |
| Reds | ['Number\_of\_differnt\_parties\_voted\_for' 'Political\_interest\_Total\_Score'  'Avg\_Satisfaction\_with\_previous\_vote' 'Most\_Important\_Issue'  'Overall\_happiness\_score' 'Avg\_size\_per\_room' 'Weighted\_education\_rank'] |
| Turquoises | ['Political\_interest\_Total\_Score' 'Avg\_Satisfaction\_with\_previous\_vote'  'Overall\_happiness\_score' 'Avg\_size\_per\_room' 'Weighted\_education\_rank'] |
| Violets | ['Avg\_Satisfaction\_with\_previous\_vote'] |
| Whites | ['Number\_of\_differnt\_parties\_voted\_for' 'Political\_interest\_Total\_Score'  'Avg\_Satisfaction\_with\_previous\_vote' 'Overall\_happiness\_score'  'Avg\_size\_per\_room' 'Weighted\_education\_rank'] |
| Yellows | ['Political\_interest\_Total\_Score' 'Avg\_Satisfaction\_with\_previous\_vote'  'Overall\_happiness\_score' 'Avg\_size\_per\_room'] |

***Manipulating the elections***

We used the following method in order to find the factor which by manipulating you are most likely to change which party will win the elections :

For each feature one by one, we add a constant value to the column, and we see if the outcome has changed or not.

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 of those modifications had 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***

Our method here is to add a constant value to each column one by one and check if the probability of winning for our coalition increased. Any change that increases the probability is kept.  
The algorithm is automatic.

We used the coalition found with the clustering, which had a probability of 69% of winning.

The method gave the following results:

***The stronger coalition gets a probability of 0.8413040527247573***

***The modified features are :***

***['Yearly\_IncomeK', 'Number\_of\_differnt\_parties\_voted\_for', 'Political\_interest\_Total\_Score', 'Avg\_Satisfaction\_with\_previous\_vote', 'Avg\_monthly\_income\_all\_years', 'Avg\_size\_per\_room', 'Weighted\_education\_rank']***

This is a much better probability than the original one.