CNBE:
An Analysis on Exit Polls

EDA & NULL VALUE CONDITION CHECK:

- Based on information of the dataset there are no null values present in the dataset, apart from 'vote' and 'gender' rest of them are all numeric. The dimension of the dataset being 1525(rows)*9(columns):

```
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   vote                    1525 non-null   object
 1   age                     1525 non-null   int64
 2   economic.cond.national  1525 non-null   int64
 3   economic.cond.household 1525 non-null   int64
 4   Blair                   1525 non-null   int64
 5   Hague                   1525 non-null   int64
 6   Europe                  1525 non-null   int64
 7   political.knowledge     1525 non-null   int64
 8   gender                  1525 non-null   object
```

- For the dataset described for numeric we could see apart from ‘age’ variable , all are in type of categorical variable represented in ‘numeric’, specifically represented of type ordinal.Scaling/Standardisation might be required depending on the model used:

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge
count	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000
mean	54.182295	3.245902	3.140328	3.334426	2.746885	6.728525	1.542295
std	15.711209	0.880969	0.929951	1.174824	1.230703	3.297538	1.083315
min	24.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000
25%	41.000000	3.000000	3.000000	2.000000	2.000000	4.000000	0.000000
50%	53.000000	3.000000	3.000000	4.000000	2.000000	6.000000	2.000000
75%	67.000000	4.000000	4.000000	4.000000	4.000000	10.000000	2.000000
max	93.000000	5.000000	5.000000	5.000000	5.000000	11.000000	3.000000

- Dataset described for type ‘object’ shows vote for ‘Labour’ party and gender ‘female’ has the most occurrence:

	vote	gender
count	1525	1525
unique	2	2
top	Labour	female
freq	1063	812

- Below info shows which party is most popular for this sampling:

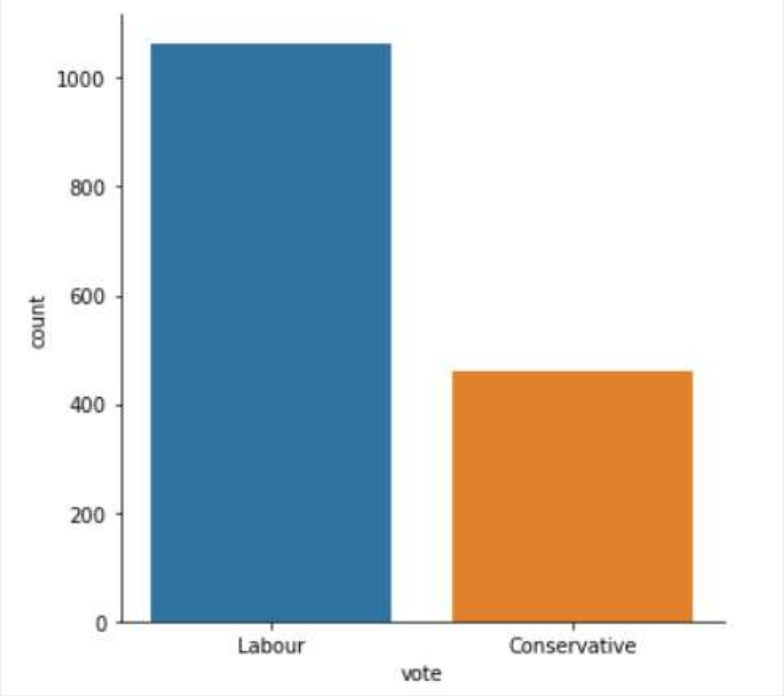
```
Labour          0.697049
Conservative    0.302951
```

- We could see about 70% of the voters sampled support ‘Labour’ party whereas only 30 percentage support ‘Conservative’.
- This uneven proportion can be of 2 reasons. One could be due to sampled population and bias during sampling (Voters in the dataset could be supporters of ‘Labour’ party).
- The other reason could be in fact the ‘Labour’ Party could be popular which in fact resembles the population

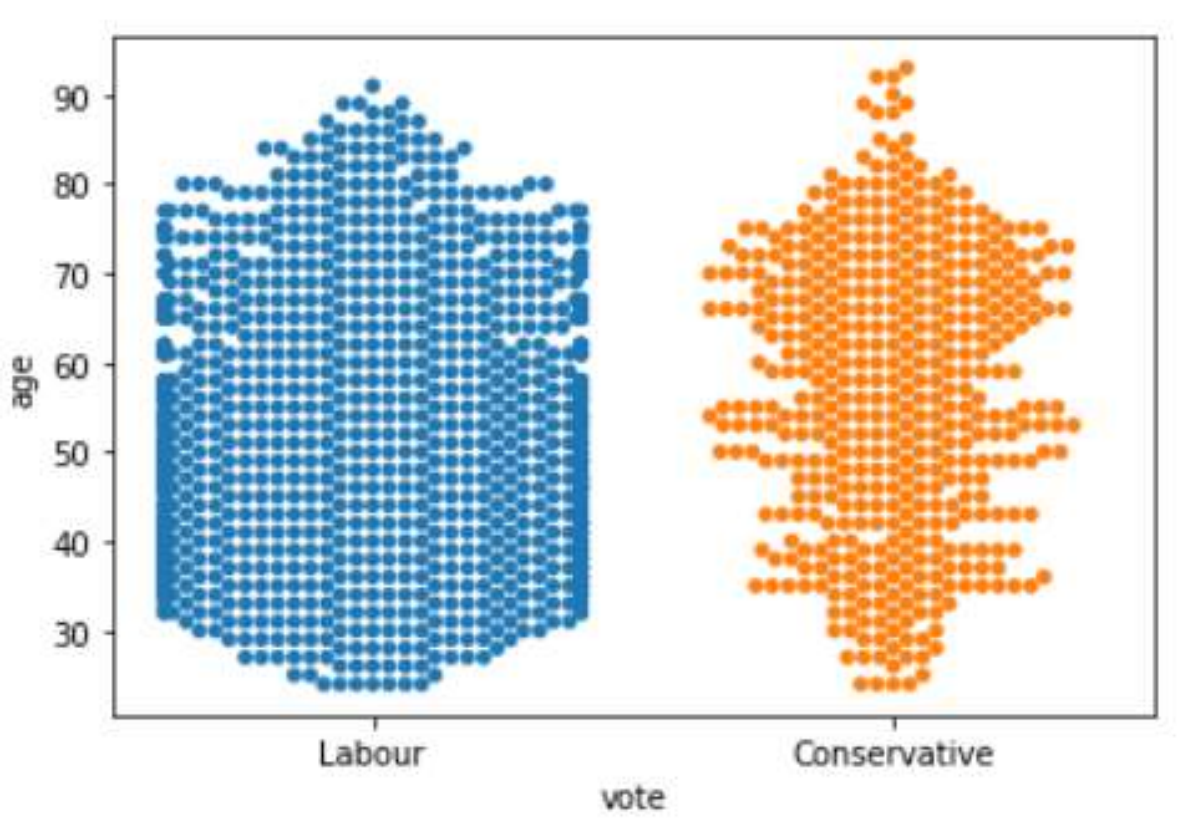
- Choice of a party within female and male is almost the same :
- ‘Labour’ party is more popular in both ‘male’ and ‘female’ category.

vote	gender	
Conservative	female	259
	male	203
Labour	female	553
	male	510

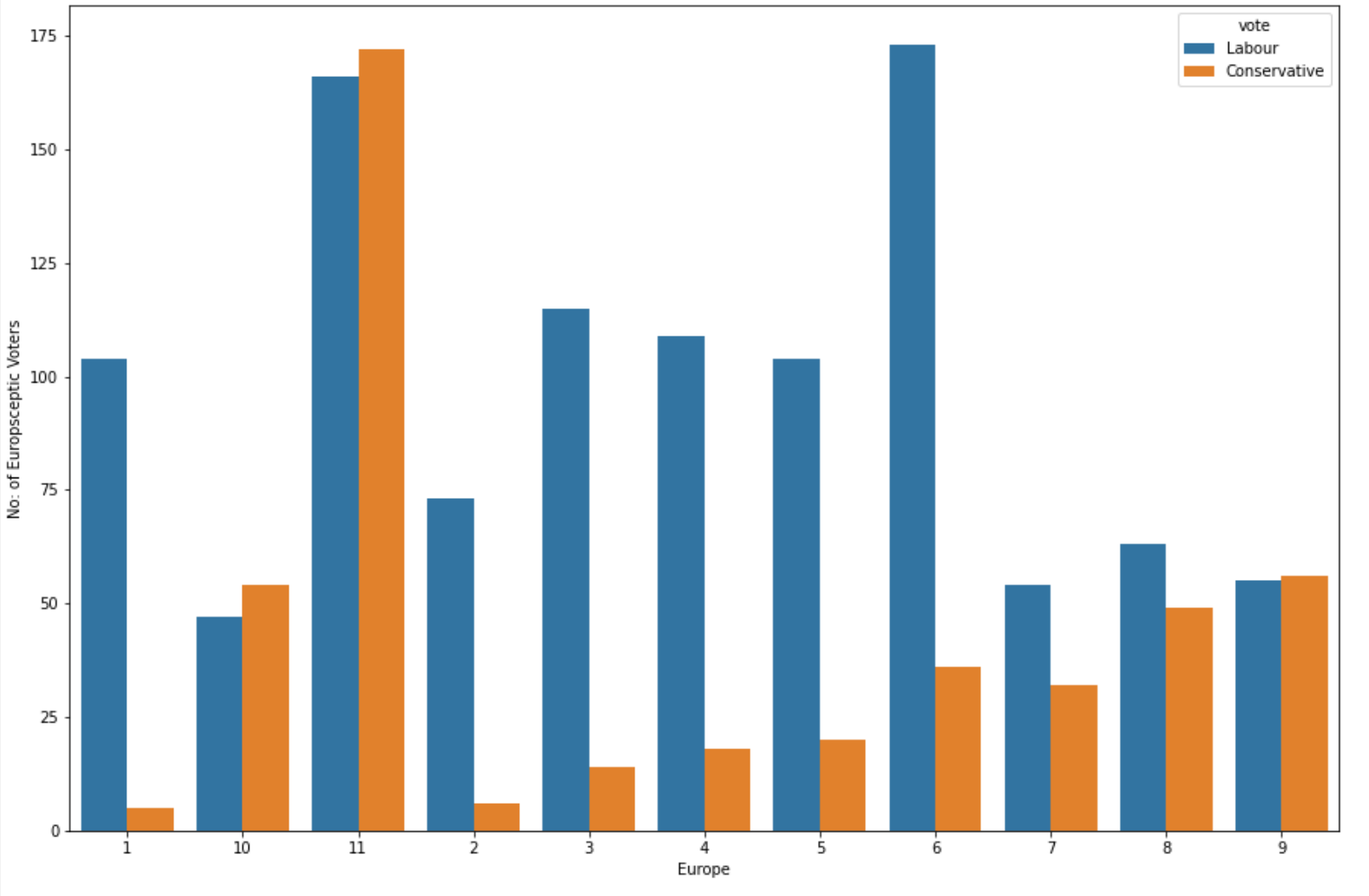
- Below visual shows popularity of the ‘Labour’ party:



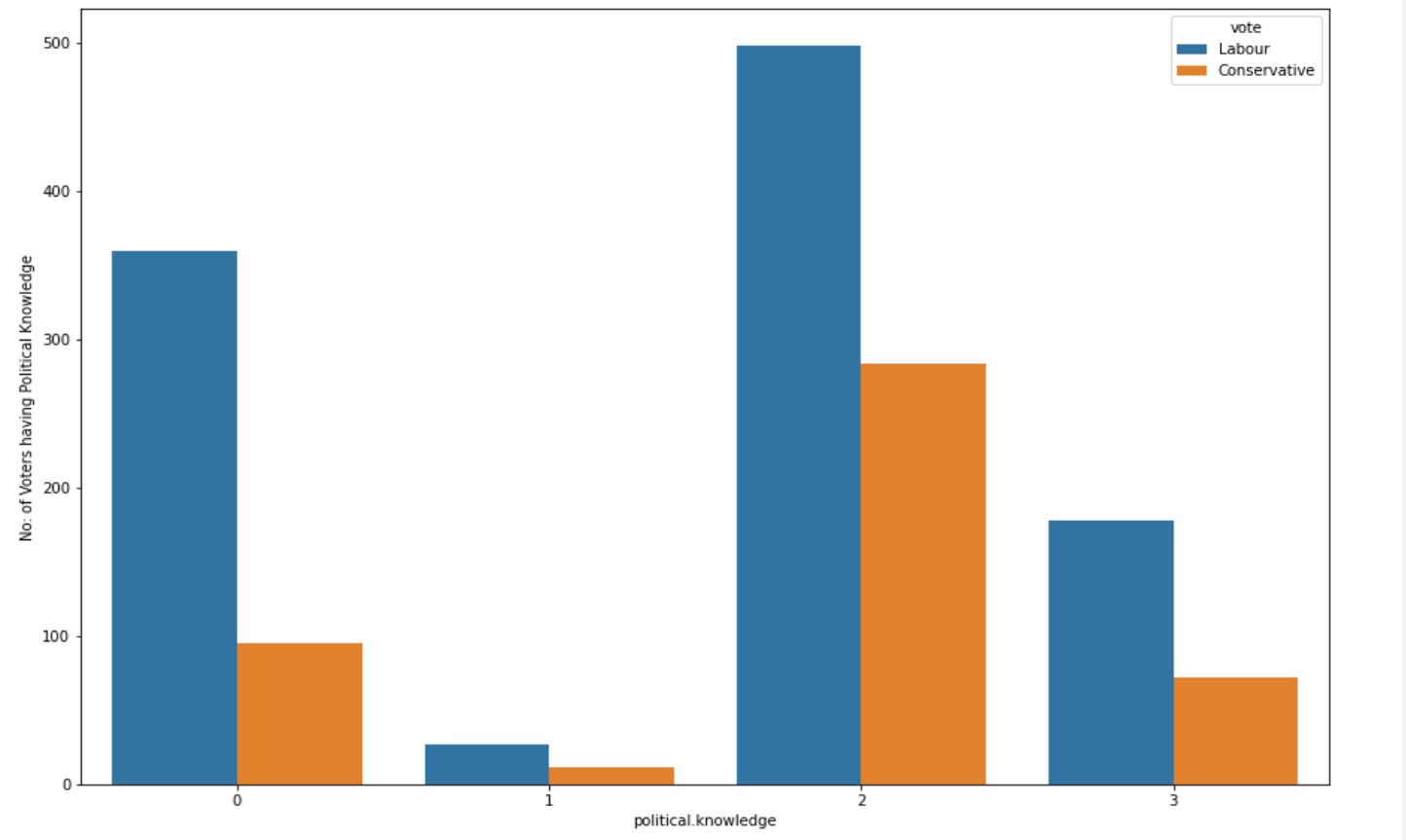
- From below we could see voter's preference is same through out different ages of the voters for 'Labour' party whereas for 'Conservative' we could see 'thinning' at various intervals .



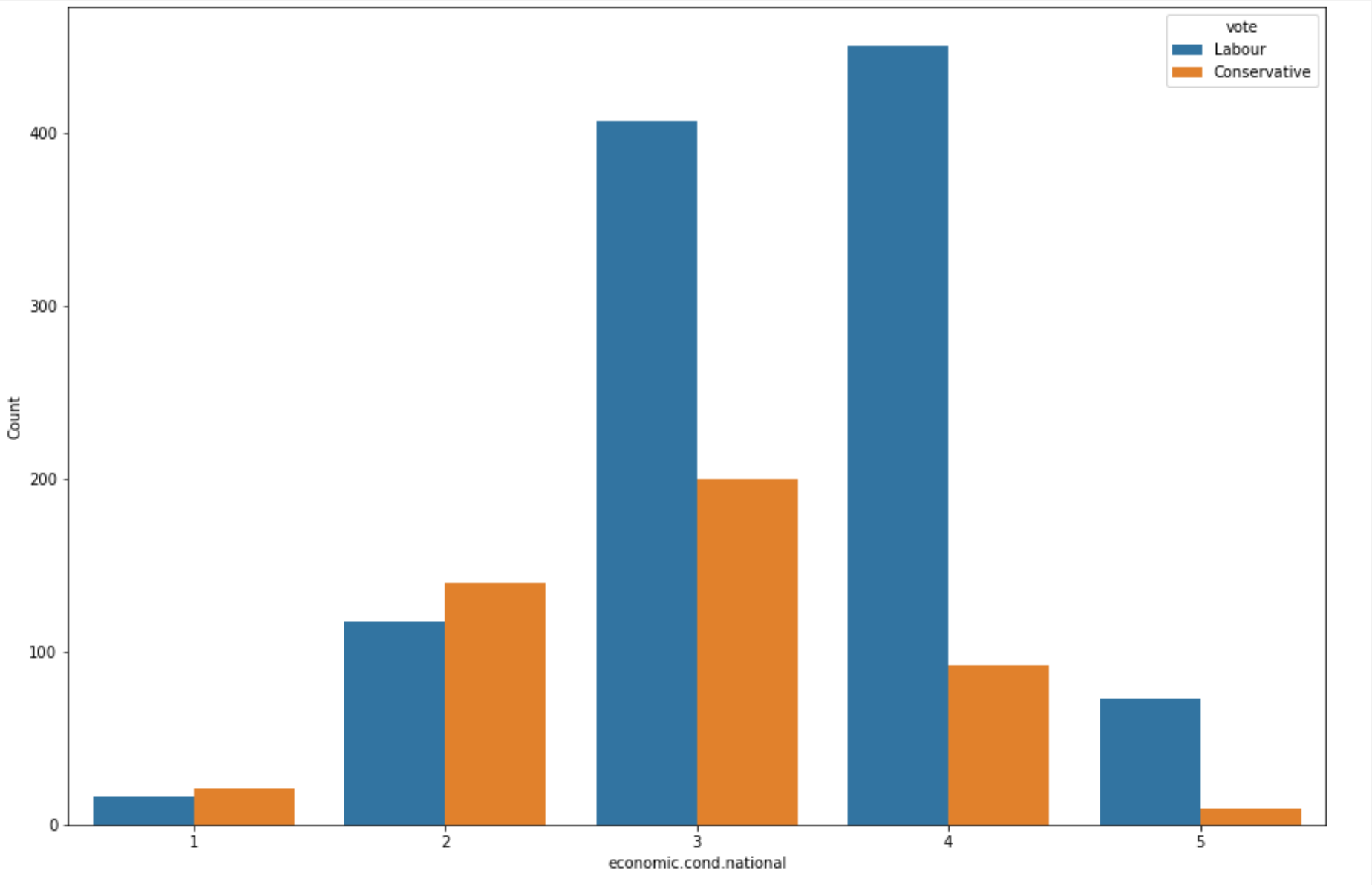
- Below visual tells us that people who oppose European integration prefer Conservative party, this trend is obvious from '9'-'11' scale .Those who support European integration prefer 'Labour' party.



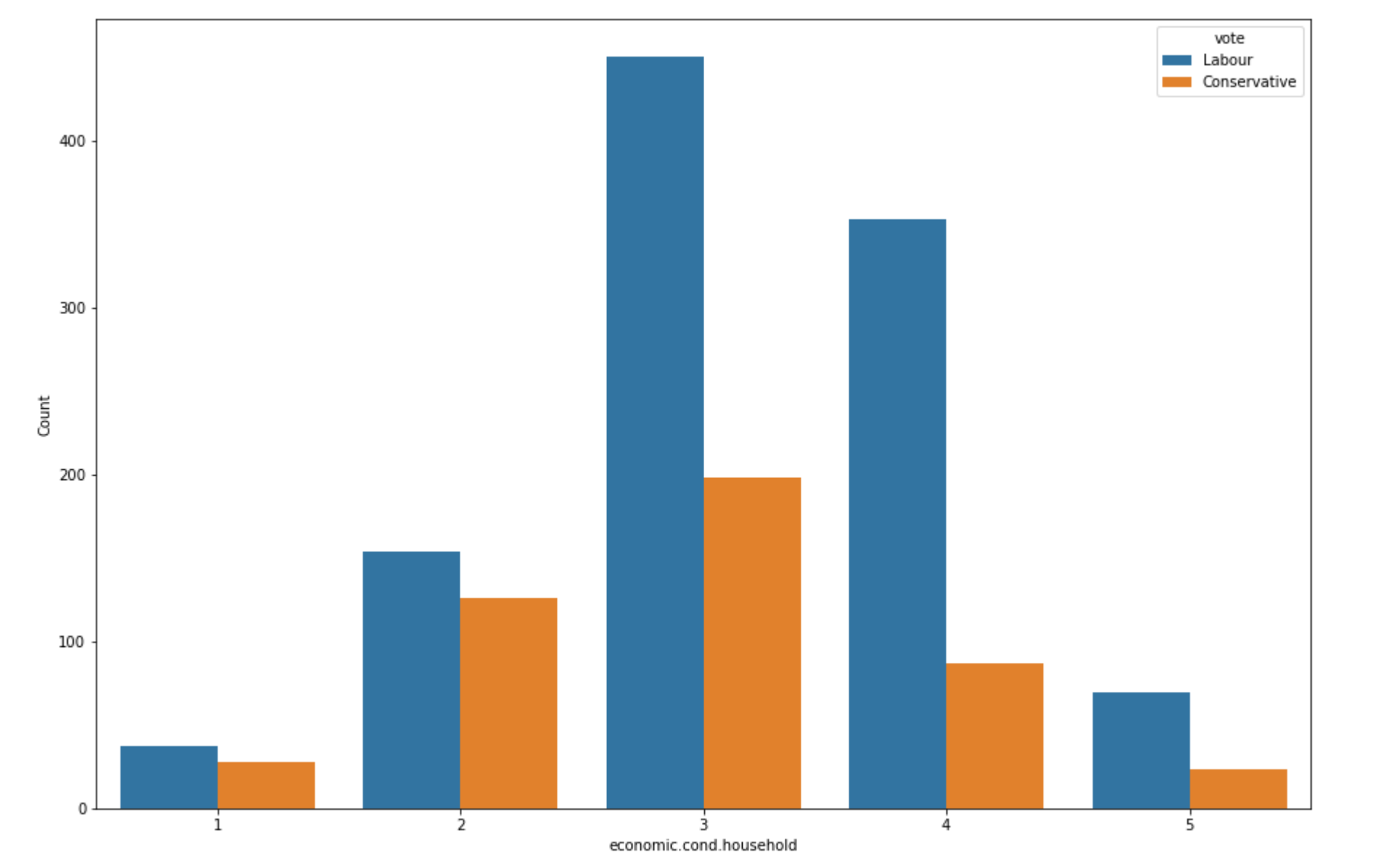
- Based on political knowledge on party's position on European integration there is no demarcation between Labour and Conservative since all the classes are dominated by Labour Party. This plot shows only one side of the information plotted with variable 'Europe'. The previous plot helped us to show the clear difference between the preference .Here, irrespective of the scale of knowledge Labour is the most popular party.



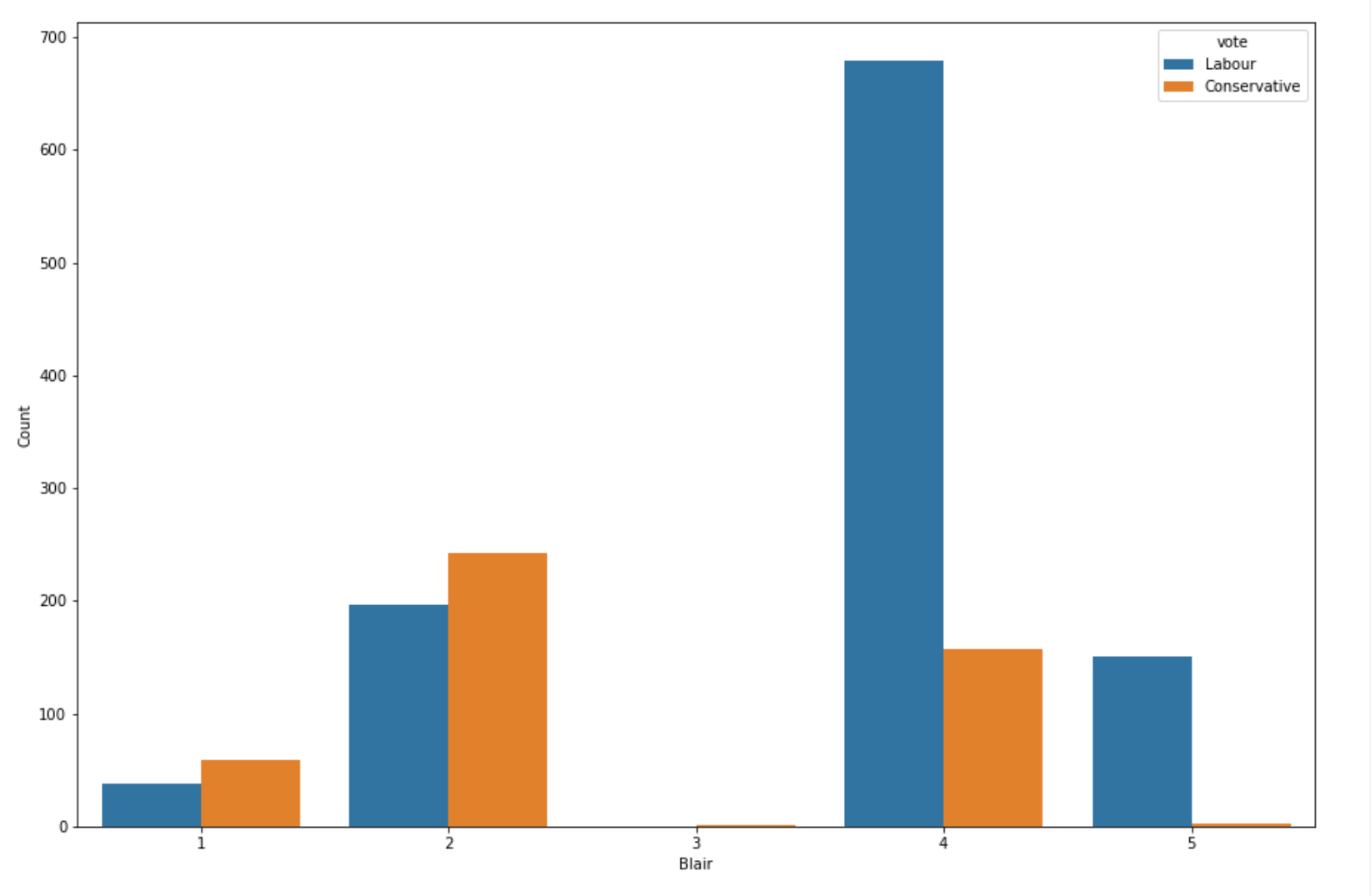
- People who want better national economic conditions prefer Labour party which is obvious from scale '3'-'5'



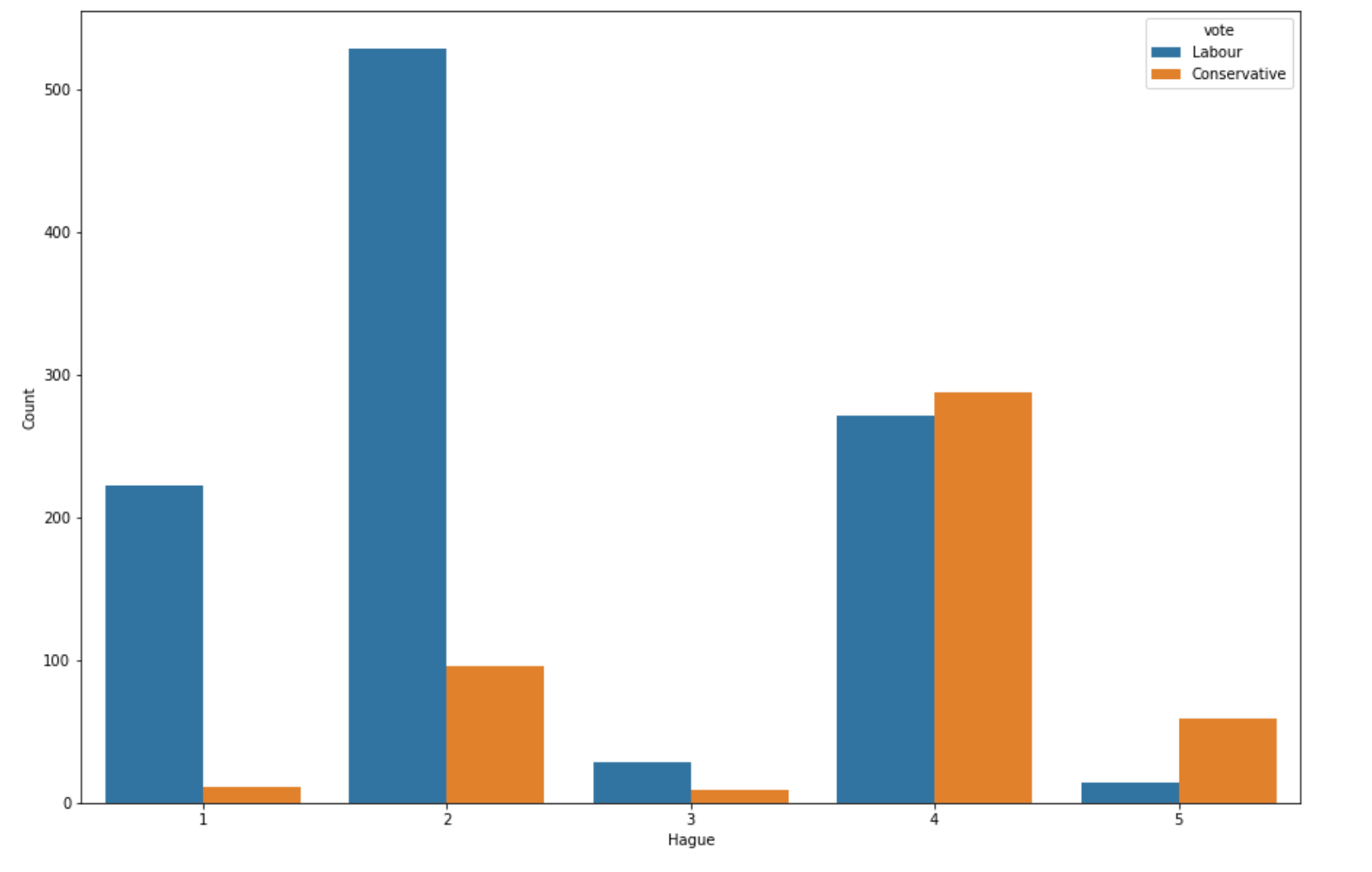
- Based on house hold economic conditions there is no demarcation between Labour party and Conservative party.



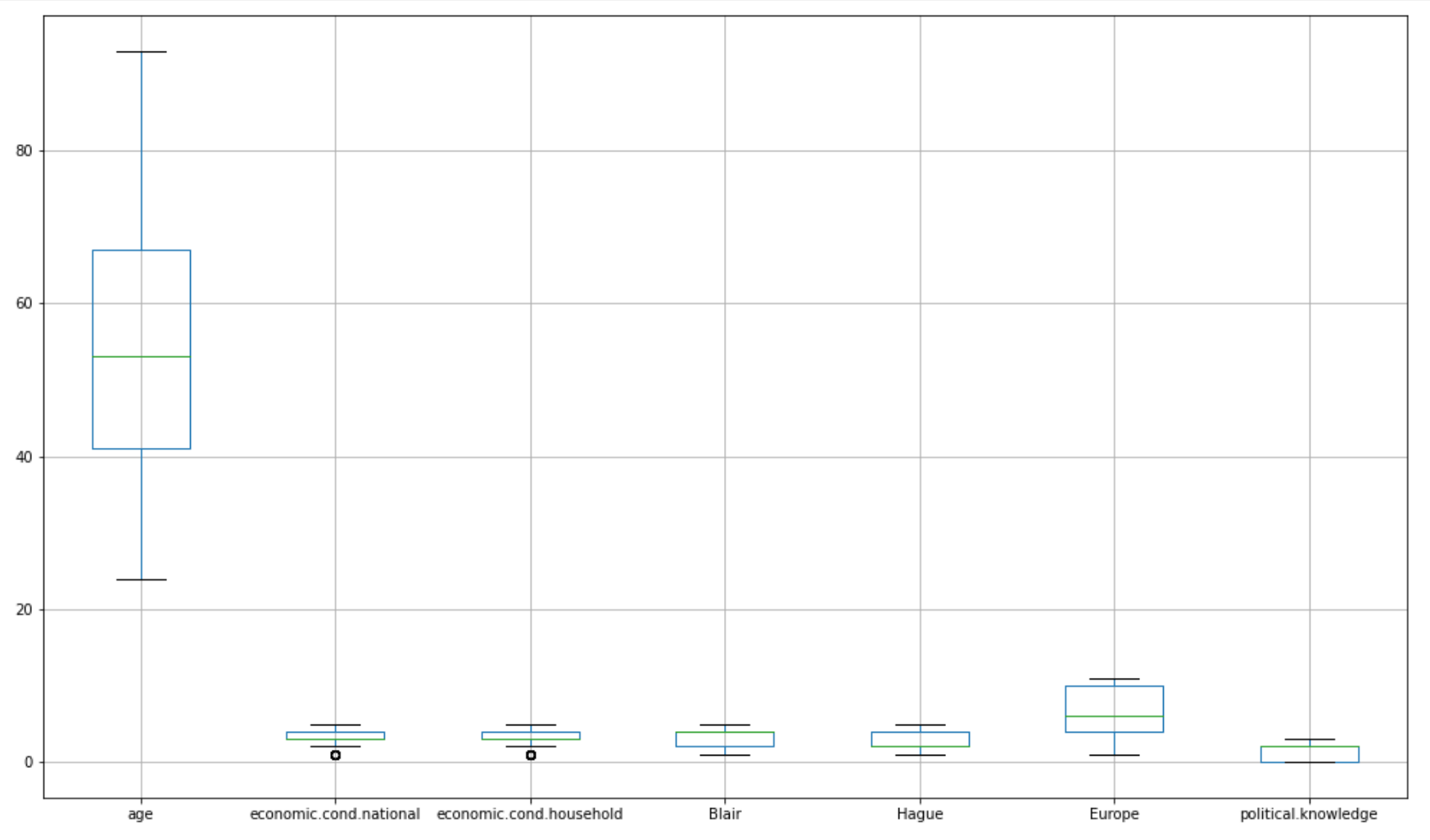
- Based on assessment of Blair, people prefer or Labour party becomes popular when assessed at scale ‘3’-’5’



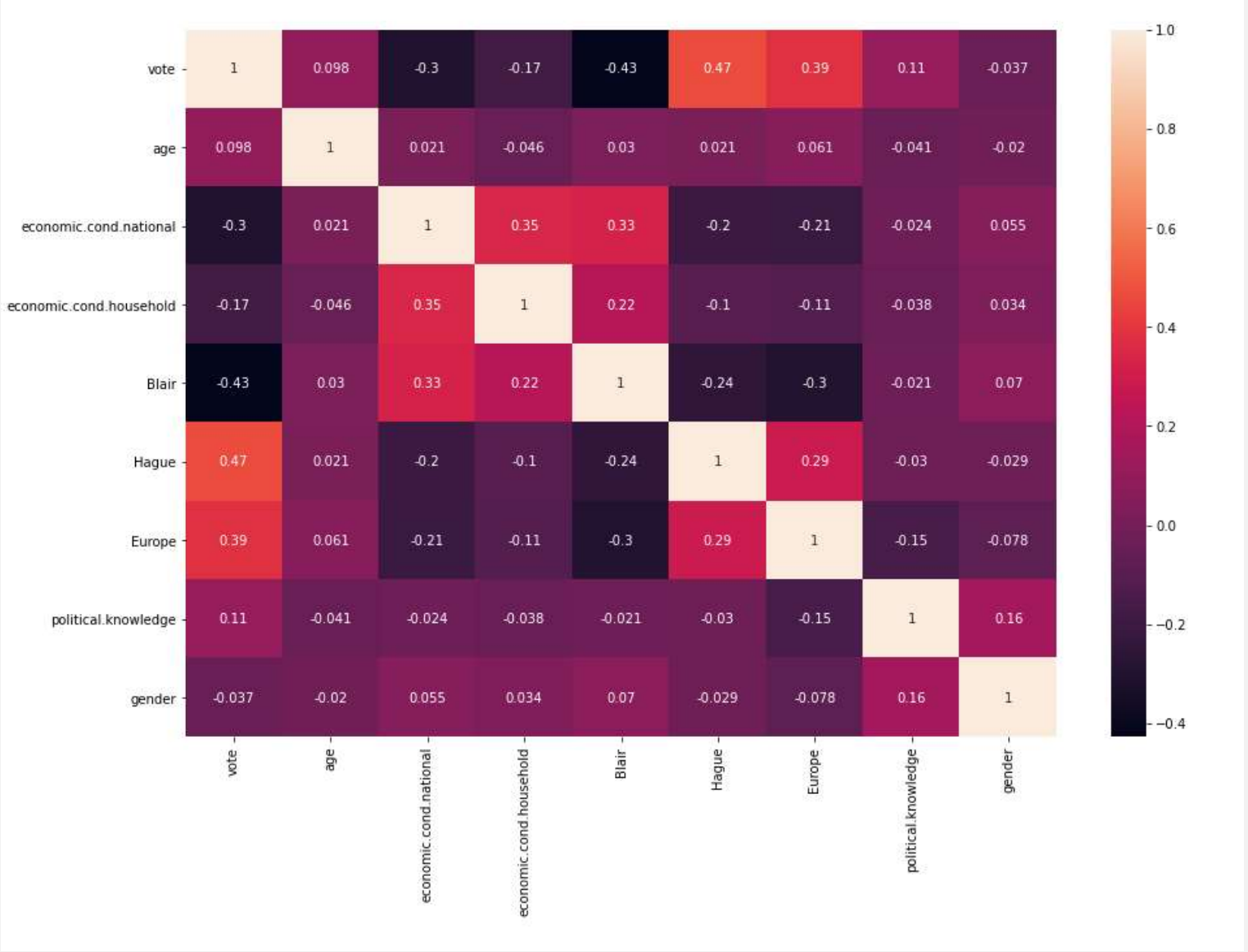
- Based on assessment of Hague , people prefer or Conservative party becomes popular when assessed at scale ‘4’ and ‘5’.



- There are no significant outliers present since all of the variables are of ordinal type which are represented in type numeric:



- From below correlation plot we could see there are no signs of multicollinearity between independent variables



ENCODING DATA, TRAIN TEST SPLIT:

- Encoding done to ‘gender’ and ‘vote’ variable
- ‘age’ variable is bin with intervals 20-40,40-60,60-80,80-100 and encoded in 0,1,2,3 resepectively
- Here we could avoid standardisation/scaling since all are in the same scale i.e. all are in same units.
- Train test split in proportions 70:30 and at random state=1.Please find more in codebook.
- Note Labour is encoded as 0 and Conservative encoded as 1.

MODEL BUILDING AND PERFORMANCE:

Linear Discrminant Analysis

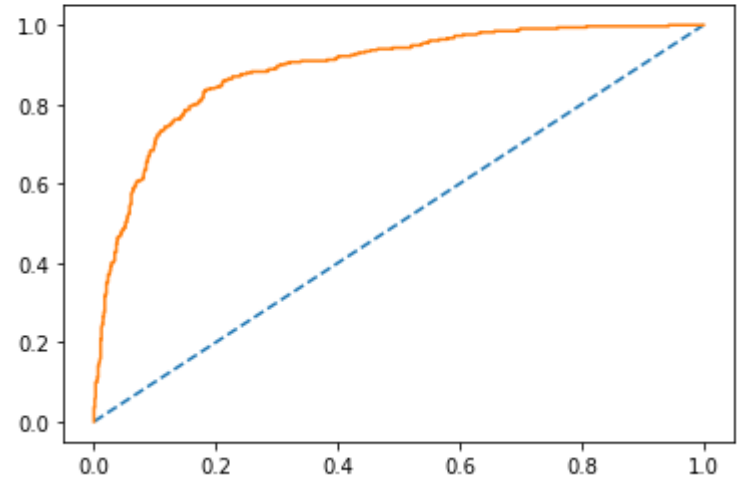
Below we could see LDA has performed well in both test and train dataset here precision and recall of '1' is low in both train and test compared class '0'.

```
Train
Accuracy score
0.8406747891283973

Confusion Matrix
[[661  74]
 [ 96 236]]

Classification Report

              precision    recall  f1-score   support

           0       0.87      0.90      0.89       735
           1       0.76      0.71      0.74       332

    accuracy                           0.84      1067
   macro avg       0.82      0.81      0.81      1067
weighted avg       0.84      0.84      0.84      1067
```

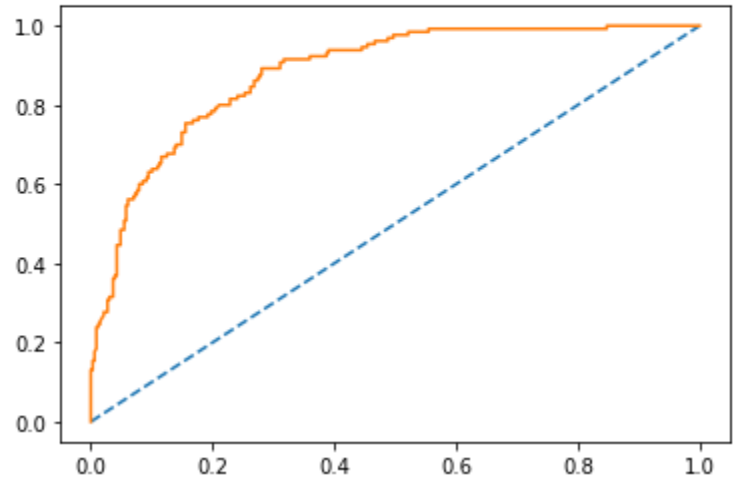
```
Test
Accuracy score
0.8231441048034934

Confusion Matrix
[[290  38]
 [ 43  87]]

Classification Report

              precision    recall  f1-score   support

           0       0.87      0.88      0.88       328
           1       0.70      0.67      0.68       130

    accuracy                           0.82       458
   macro avg       0.78      0.78      0.78       458
weighted avg       0.82      0.82      0.82       458
```

ROC-AUC has good score for both train and test. LDA has done appreciably well in identifying points in test case

```
Train
AUC: 0.889

[<matplotlib.lines.Line2D at 0x1828f6bab38>]
```




```
Test
AUC: 0.882

[<matplotlib.lines.Line2D at 0x1828f34ba90>]
```



Logistic Regression

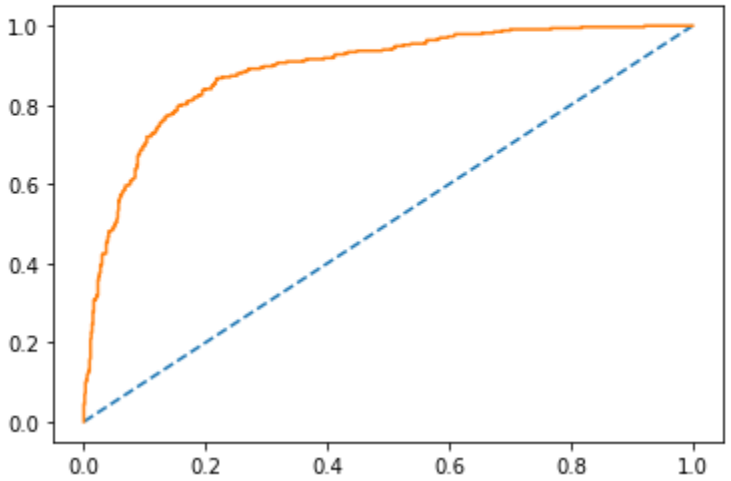
Below we could see Logistic regression has performed well in both test and train dataset here precision and recall of '1' is low in both train and test compared class '0'.

```
Train
Accuracy score
0.8369259606373008

Confusion Matrix
[[664  71]
 [103 229]]

Classification Report

              precision    recall  f1-score   support

           0       0.87      0.90      0.88       735
           1       0.76      0.69      0.72       332

    accuracy                           0.84      1067
   macro avg       0.81      0.80      0.80      1067
weighted avg       0.83      0.84      0.83      1067
```

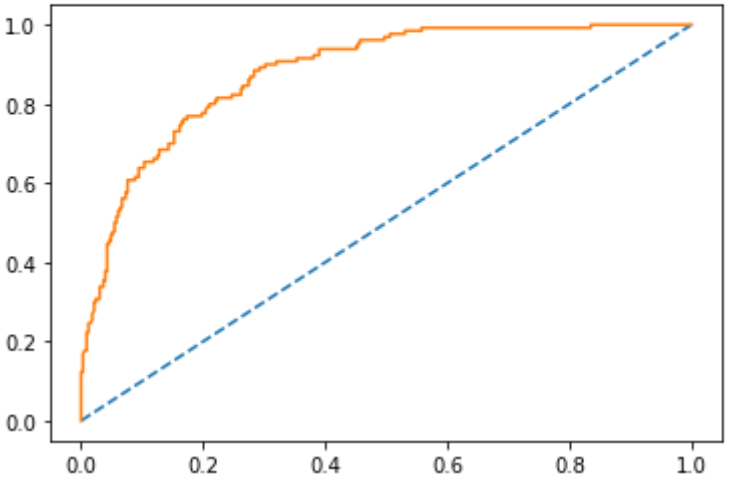
```
Test
Accuracy score
0.8209606986899564

Confusion Matrix
[[291  37]
 [ 45  85]]

Classification Report

              precision    recall  f1-score   support

           0       0.87      0.89      0.88       328
           1       0.70      0.65      0.67       130

    accuracy                           0.82       458
   macro avg       0.78      0.77      0.78       458
weighted avg       0.82      0.82      0.82       458
```

ROC-AUC has good score for both train and test. Logistic regression has done appreciably well in identifying points in test case also.

```
Train
AUC: 0.889

[<matplotlib.lines.Line2D at 0x1828f39e5c0>]
```




```
Test
AUC: 0.881

[<matplotlib.lines.Line2D at 0x1828fd96198>]
```



K Nearest Neighbours

Below we could see KNN has performed well in both train dataset but there is a 6% drop in accuracy in test dataset , here precision and recall of '1' is low in both train and test compared class '0'.

Train
Accuracy score
0.8687910028116214

Confusion Matrix
[[672 63]
 [77 255]]

Classification Report

	precision	recall	f1-score	support
0	0.90	0.91	0.91	735
1	0.80	0.77	0.78	332
accuracy			0.87	1067
macro avg	0.85	0.84	0.85	1067
weighted avg	0.87	0.87	0.87	1067

Test
Accuracy score
0.8013100436681223

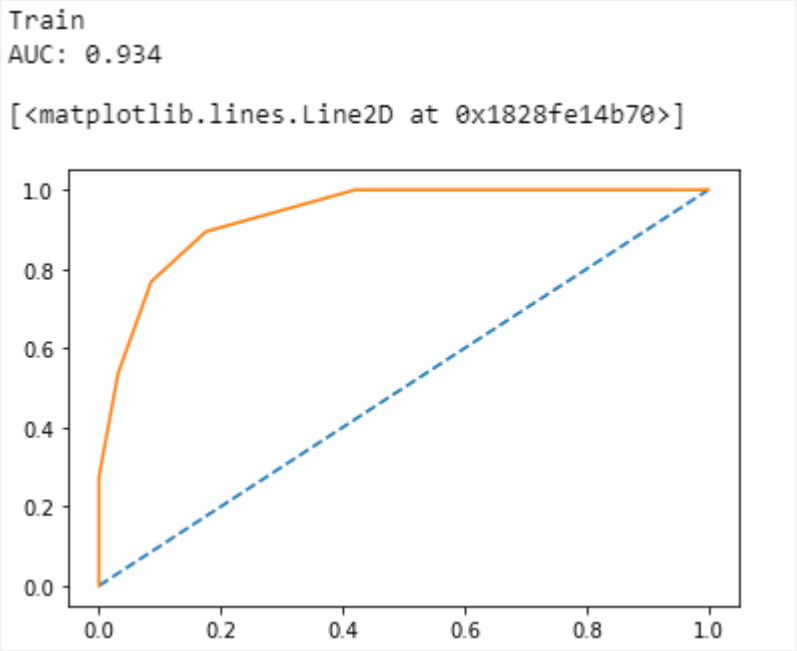
Confusion Matrix
[[278 50]
 [41 89]]

Classification Report

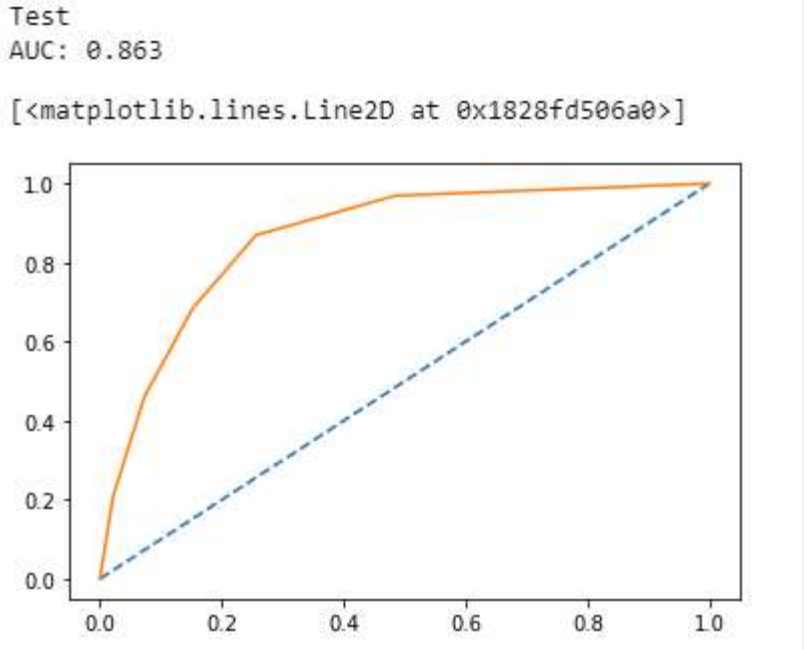
	precision	recall	f1-score	support
0	0.87	0.85	0.86	328
1	0.64	0.68	0.66	130
accuracy			0.80	458
macro avg	0.76	0.77	0.76	458
weighted avg	0.81	0.80	0.80	458

ROC-AUC has good score for both train and test. KNN has done appreciably well in identifying points in test case also(a drop of 7% in AUC can be seen in test).

```
Train
AUC: 0.934

[<matplotlib.lines.Line2D at 0x1828fe14b70>]
```




```
Test
AUC: 0.863

[<matplotlib.lines.Line2D at 0x1828fd506a0>]
```



Gaussian Naïve Bayes

Below we could see GaussianNB has performed well in both train dataset and test dataset , here precision and recall of '1' is low in both train and test compared class '0'.

Train
Accuracy score
0.8341143392689785

Confusion Matrix
[[649 86]
 [91 241]]

Classification Report

	precision	recall	f1-score	support
0	0.88	0.88	0.88	735
1	0.74	0.73	0.73	332
accuracy			0.83	1067
macro avg	0.81	0.80	0.81	1067
weighted avg	0.83	0.83	0.83	1067

Test
Accuracy score
0.8275109170305677

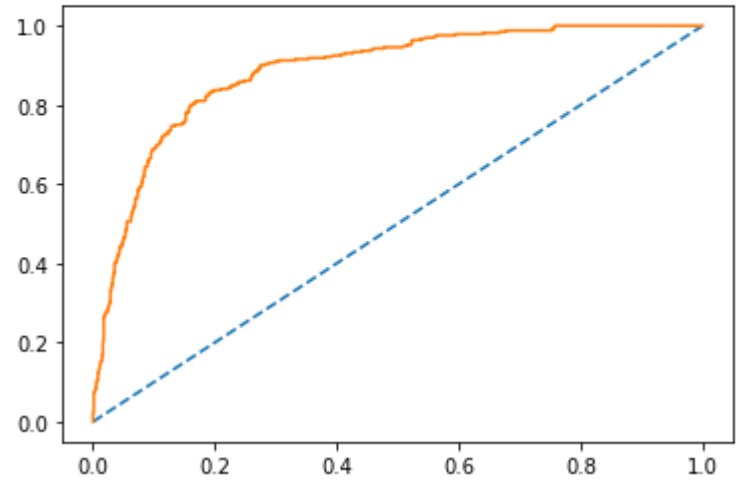
Confusion Matrix
[[285 43]
 [36 94]]

Classification Report

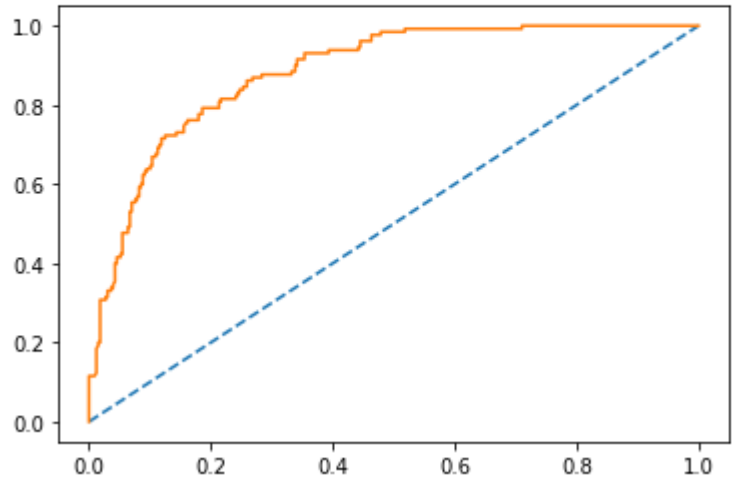
	precision	recall	f1-score	support
0	0.89	0.87	0.88	328
1	0.69	0.72	0.70	130
accuracy			0.83	458
macro avg	0.79	0.80	0.79	458
weighted avg	0.83	0.83	0.83	458

ROC-AUC has good score for both train and test. Gaussian NB has done appreciably well in identifying points in test case also.

```
Train
AUC: 0.886

[<matplotlib.lines.Line2D at 0x1828f6f14e0>]
```




```
Test
AUC: 0.883

[<matplotlib.lines.Line2D at 0x1828ff31d68>]
```



Support Vector Machine

Below we could see SVM has performed well in both train dataset but there is a 3% drop in accuracy in test dataset , here precision and recall of '1' is low in both train and test compared to class '0'.

Train
Accuracy score
0.8425492033739457

Confusion Matrix
[[677 58]
 [110 222]]

Classification Report

	precision	recall	f1-score	support
0	0.86	0.92	0.89	735
1	0.79	0.67	0.73	332
accuracy			0.84	1067
macro avg	0.83	0.79	0.81	1067
weighted avg	0.84	0.84	0.84	1067

Test
Accuracy score
0.8165938864628821

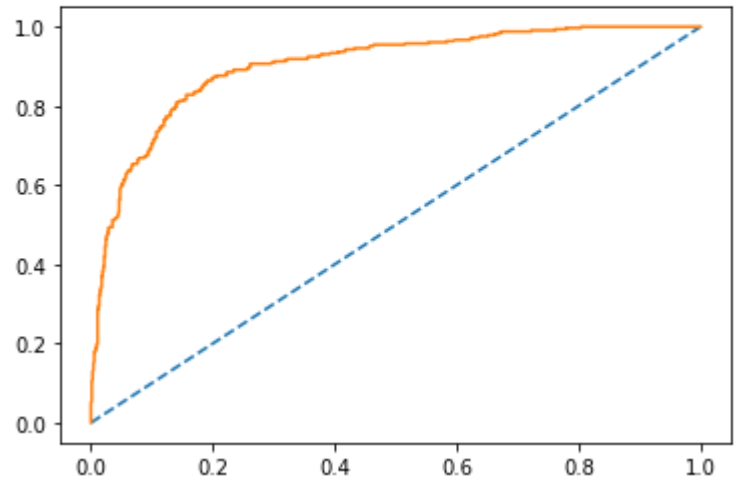
Confusion Matrix
[[289 39]
 [45 85]]

Classification Report

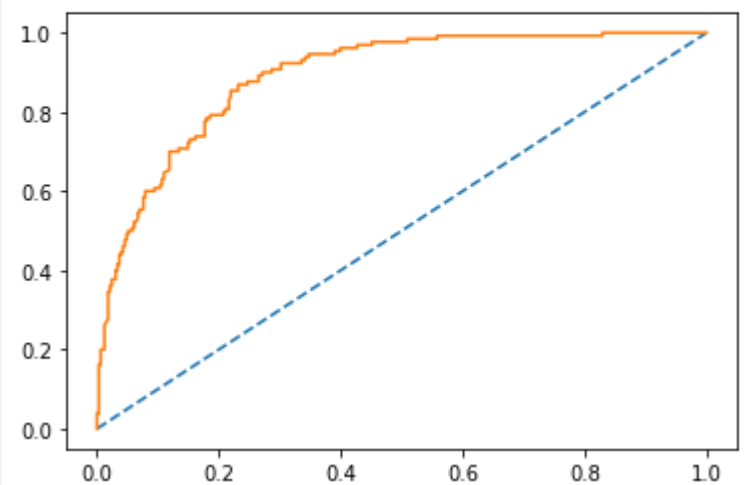
	precision	recall	f1-score	support
0	0.87	0.88	0.87	328
1	0.69	0.65	0.67	130
accuracy			0.82	458
macro avg	0.78	0.77	0.77	458
weighted avg	0.81	0.82	0.82	458

ROC-AUC has good score for both train and test. SVM has done appreciably well in identifying points in test case also.

```
Train
AUC: 0.902

[<matplotlib.lines.Line2D at 0x18290f617b8>]
```




```
Test
AUC: 0.891

[<matplotlib.lines.Line2D at 0x1828fe317b8>]
```



MODEL TUNING , BAGGING AND BOOSTING

- SMOTE is a technique used to enhance model performance , here we would not be using SMOTE.
- SMOTE is used to artificially create data to treat the imbalance of the target variable .This could affect the characteristics of the data in hand.
- Moreover the data is regarding exit-poll prediction , using SMOTE here would not make much sense.
- Therefore we could use GridSearch CV method to tune the parameters of model.
- Data tuning was partly done in Data Encoding section when we binned the 'age' variable to make it numeric.

COMPARISON OF MODEL PERFORMANCE:

- Even after tuning there has been no significant increase in model performance, overall all measures remains same as before tuning:

Before Tuning

Train

	LDA_Train	Logistic_Train	KNN_Train	GaussianNb_Train	SVM_Train
Accuracy_train	0.84	0.84	0.87	0.83	0.84
AUC_train	0.89	0.89	0.93	0.89	0.90
Recall_Labour_train	0.90	0.90	0.91	0.88	0.92
Precision_Labour_train	0.87	0.87	0.90	0.88	0.86
Recall_Conser_train	0.71	0.69	0.77	0.73	0.67
Precision_Conser_train	0.90	0.90	0.91	0.88	0.92

Test

	LDA_test	Logistic_test	KNN_test	GaussianNb_test	SVM_test
Accuracy_test	0.82	0.82	0.80	0.83	0.82
AUC_test	0.88	0.88	0.86	0.88	0.89
Recall_Labour_test	0.88	0.89	0.85	0.87	0.88
Precision_Labour_test	0.87	0.87	0.87	0.89	0.87
Recall_Conser_test	0.67	0.65	0.68	0.72	0.65
Precision_Conser_test	0.88	0.89	0.85	0.87	0.88

After Tuning

Train

	LDA_tune_Train	Logistic_tune_Train	KNN_tune_Train	GaussianNb_Train	SVM_tune_Train
Accuracy_train	0.84	0.83	0.87	0.83	0.84
AUC_train	0.89	0.89	0.93	0.89	0.89
Recall_Labour_train	0.90	0.92	0.91	0.88	0.90
Precision_Labour_train	0.87	0.85	0.90	0.88	0.87
Recall_Conser_train	0.71	0.64	0.77	0.73	0.70
Precision_Conser_train	0.90	0.92	0.91	0.88	0.90

Test

	LDA_test	Logistic_test	KNN_test	GaussianNb_test	SVM_test
Accuracy_test	0.82	0.82	0.80	0.83	0.82
AUC_test	0.88	0.88	0.86	0.88	0.89
Recall_Labour_test	0.88	0.89	0.85	0.87	0.88
Precision_Labour_test	0.87	0.87	0.87	0.89	0.87
Recall_Conser_test	0.67	0.65	0.68	0.72	0.65
Precision_Conser_test	0.88	0.89	0.85	0.87	0.88

BAGGING AND BOOSTING:

Bagging Classifier:

For train set Bagging classifier performs well while there is significant percentage drop in accuracy , recall and precision.

Train
Accuracy score
0.9531396438612934

Confusion Matrix
[[719 16]
 [34 298]]

Classification Report

	precision	recall	f1-score	support
0	0.95	0.98	0.97	735
1	0.95	0.90	0.92	332
accuracy			0.95	1067
macro avg	0.95	0.94	0.94	1067
weighted avg	0.95	0.95	0.95	1067

Test
Accuracy score
0.8144104803493449

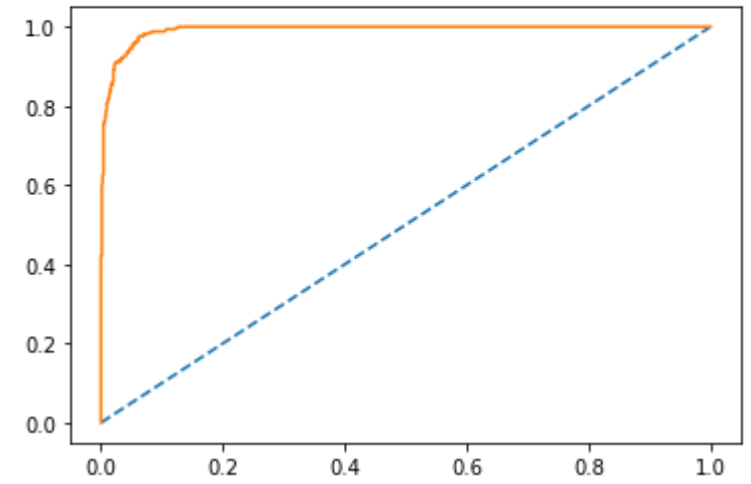
Confusion Matrix
[[285 43]
 [42 88]]

Classification Report

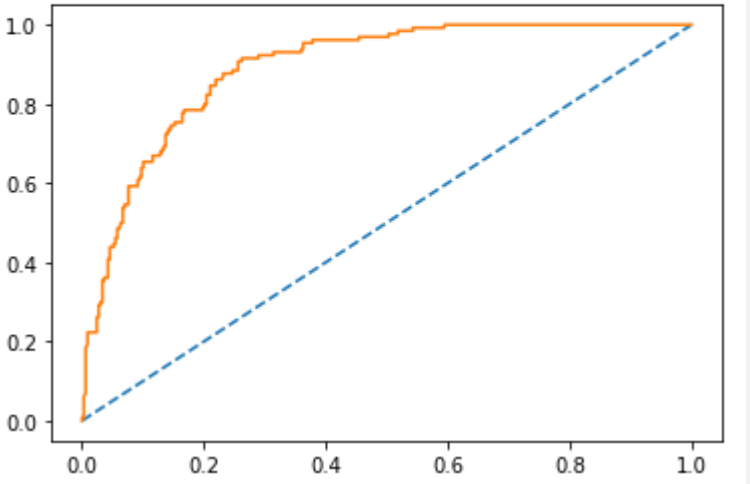
	precision	recall	f1-score	support
0	0.87	0.87	0.87	328
1	0.67	0.68	0.67	130
accuracy			0.81	458
macro avg	0.77	0.77	0.77	458
weighted avg	0.81	0.81	0.81	458

- Similarly Bagging classifier seems to overfit on train set which is reflected on test set we could see 10% drop in AUC score

```
Train
AUC: 0.991

[<matplotlib.lines.Line2D at 0x18291e73ba8>]
```




```
Test
AUC: 0.891

[<matplotlib.lines.Line2D at 0x1828fd30860>]
```



BAGGING AND BOOSTING:

XGBoost Classifier:

XGBoost does well on train set but all scores diminishes significantly in test set.

Train
Accuracy score
0.9662605435801312

Confusion Matrix
[[721 14]
 [22 310]]

Classification Report

	precision	recall	f1-score	support
0	0.97	0.98	0.98	735
1	0.96	0.93	0.95	332
accuracy			0.97	1067
macro avg	0.96	0.96	0.96	1067
weighted avg	0.97	0.97	0.97	1067

Test
Accuracy score
0.8056768558951966

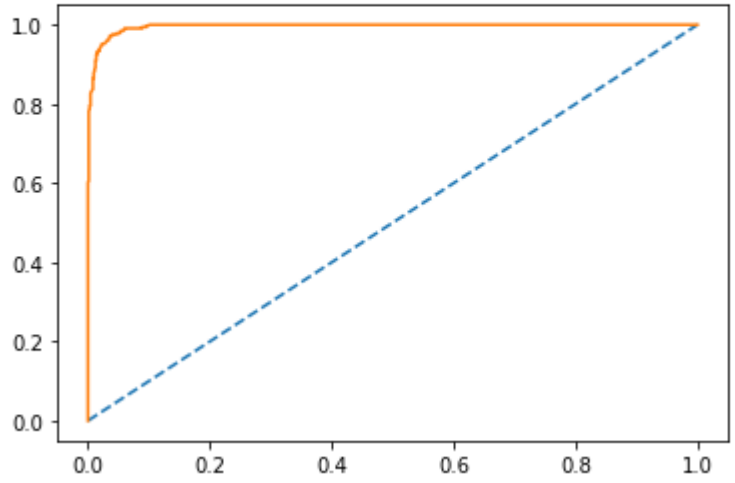
Confusion Matrix
[[277 51]
 [38 92]]

Classification Report

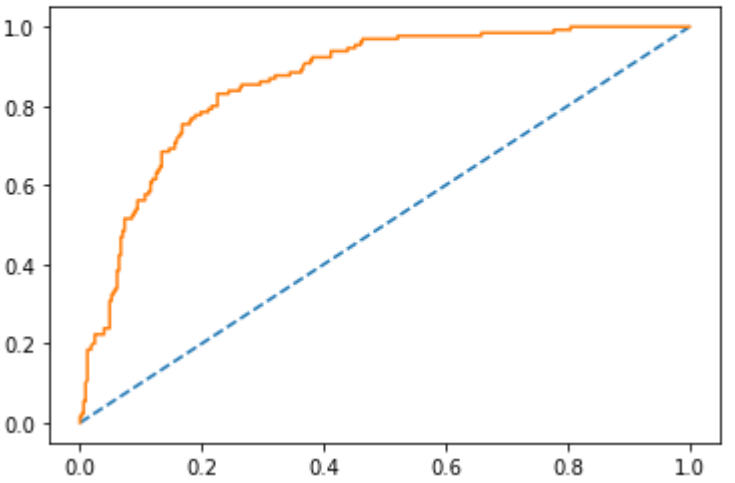
	precision	recall	f1-score	support
0	0.88	0.84	0.86	328
1	0.64	0.71	0.67	130
accuracy			0.81	458
macro avg	0.76	0.78	0.77	458
weighted avg	0.81	0.81	0.81	458

- A similar trend could be observed ROC-AUC score also.

```
Train
AUC: 0.996

[<matplotlib.lines.Line2D at 0x18292009fd0>]
```




```
Test
AUC: 0.862

[<matplotlib.lines.Line2D at 0x18291e73f60>]
```



BAGGING AND BOOSTING:

AdaBoost Classifier:

AdaBoost performs equally well in both train and test set.

Train
Accuracy score
0.8406747891283973

Confusion Matrix
[[662 73]
 [97 235]]

Classification Report

	precision	recall	f1-score	support
0	0.87	0.90	0.89	735
1	0.76	0.71	0.73	332
accuracy			0.84	1067
macro avg	0.82	0.80	0.81	1067
weighted avg	0.84	0.84	0.84	1067

Test
Accuracy score
0.8144104803493449

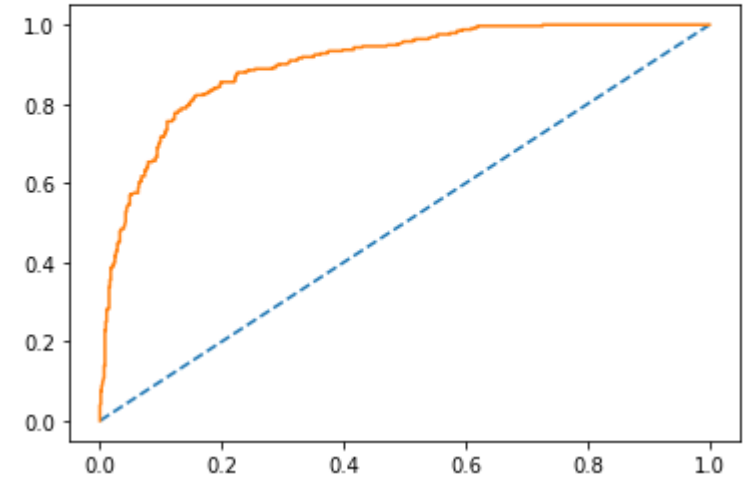
Confusion Matrix
[[285 43]
 [42 88]]

Classification Report

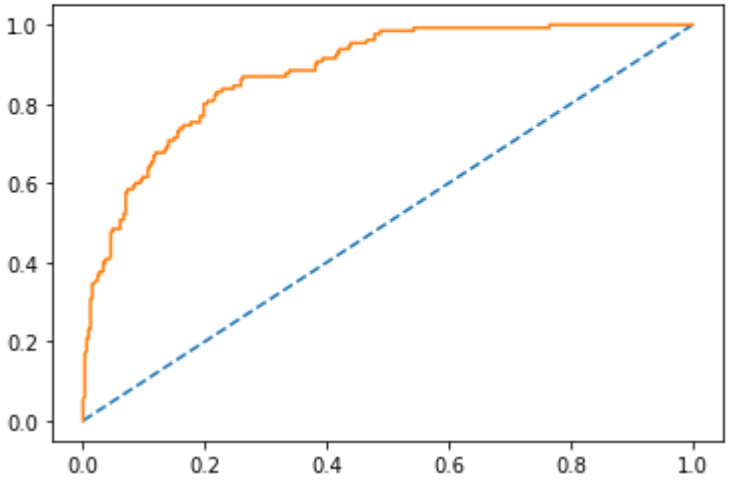
	precision	recall	f1-score	support
0	0.87	0.87	0.87	328
1	0.67	0.68	0.67	130
accuracy			0.81	458
macro avg	0.77	0.77	0.77	458
weighted avg	0.81	0.81	0.81	458

- Also for ROC-AUC score AdaBoost classifier performs equally well in both train and test set.

```
Train
AUC: 0.901

[<matplotlib.lines.Line2D at 0x18291af87b8>]
```




```
Test
AUC: 0.881

[<matplotlib.lines.Line2D at 0x18291e8e978>]
```



- Comparing train and test set of XGBoost , Bagging and AdaBoost . All performs equally well .Bagging Classifier performs well in test set in terms of AUC score , AdaBoost and Bagging performs well for test set in terms of Recall(Labour) and Precision(Conservative) while XGBoost performs well for Recall(Conservative)

Train

	Bagging_Train	XGBoost_Train	AdaBoost_Train
Accuracy_train	0.95	0.97	0.84
AUC_train	0.99	1.00	0.90
Recall_Labour_train	0.98	0.98	0.90
Precision_Labour_train	0.95	0.97	0.87
Recall_Conser_train	0.90	0.93	0.71
Precision_Conser_train	0.98	0.98	0.90

Test

	Bagging_test	XGBoost_test	AdaBoost_test
Accuracy_test	0.81	0.81	0.81
AUC_test	0.89	0.86	0.88
Recall_Labour_test	0.87	0.84	0.87
Precision_Labour_test	0.87	0.88	0.87
Recall_Conser_test	0.68	0.71	0.68
Precision_Conser_test	0.87	0.84	0.87

COMPARISON OF DIFFERENT MODELS BEFORE TUNING:

Train

	Bagging_Train	XGBoost_Train	AdaBoost_Train	GBoost_Train
Accuracy_train	0.95	0.97	0.84	0.88
AUC_train	0.99	1.00	0.90	0.94
Recall_Labour_train	0.98	0.98	0.90	0.92
Precision_Labour_train	0.95	0.97	0.87	0.90
Recall_Conser_train	0.90	0.93	0.71	0.78
Precision_Conser_train	0.98	0.98	0.90	0.92

Test

	Bagging_test	XGBoost_test	AdaBoost_test	GBoost_test
Accuracy_test	0.81	0.81	0.81	0.82
AUC_test	0.89	0.86	0.88	0.90
Recall_Labour_test	0.87	0.84	0.87	0.86
Precision_Labour_test	0.87	0.88	0.87	0.89
Recall_Conser_test	0.68	0.71	0.68	0.72
Precision_Conser_test	0.87	0.84	0.87	0.86

	LDA_Train	Logistic_Train	KNN_Train	GaussianNb_Train	SVM_Train
Accuracy_train	0.84	0.84	0.87	0.83	0.84
AUC_train	0.89	0.89	0.93	0.89	0.90
Recall_Labour_train	0.90	0.90	0.91	0.88	0.92
Precision_Labour_train	0.87	0.87	0.90	0.88	0.86
Recall_Conser_train	0.71	0.69	0.77	0.73	0.67
Precision_Conser_train	0.90	0.90	0.91	0.88	0.92

	LDA_test	Logistic_test	KNN_test	GaussianNb_test	SVM_test
Accuracy_test	0.82	0.82	0.80	0.83	0.82
AUC_test	0.88	0.88	0.86	0.88	0.89
Recall_Labour_test	0.88	0.89	0.85	0.87	0.88
Precision_Labour_test	0.87	0.87	0.87	0.89	0.87
Recall_Conser_test	0.67	0.65	0.68	0.72	0.65
Precision_Conser_test	0.88	0.89	0.85	0.87	0.88

COMPARISON OF DIFFERENT MODELS AFTER TUNING:

Train

	Bagging_Train	XGBoost_Train	AdaBoost_Train	GBoost_Train
Accuracy_train	0.93	0.88	0.84	0.91
AUC_train	0.98	0.95	0.90	0.96
Recall_Labour_train	0.96	0.93	0.90	0.95
Precision_Labour_train	0.94	0.90	0.87	0.92
Recall_Conser_train	0.87	0.78	0.71	0.83
Precision_Conser_train	0.96	0.93	0.90	0.95

	LDA_tune_Train	Logistic_tune_Train	KNN_tune_Train	GaussianNb_Train	SVM_tune_Train
Accuracy_train	0.84	0.83	0.87	0.83	0.84
AUC_train	0.89	0.89	0.93	0.89	0.89
Recall_Labour_train	0.90	0.92	0.91	0.88	0.90
Precision_Labour_train	0.87	0.85	0.90	0.88	0.87
Recall_Conser_train	0.71	0.64	0.77	0.73	0.70
Precision_Conser_train	0.90	0.92	0.91	0.88	0.90

Test

	Bagging_test	XGBoost_test	AdaBoost_test	GBoost_test
Accuracy_test	0.83	0.82	0.82	0.81
AUC_test	0.90	0.89	0.88	0.89
Recall_Labour_test	0.88	0.86	0.87	0.86
Precision_Labour_test	0.88	0.88	0.87	0.88
Recall_Conser_test	0.71	0.72	0.68	0.71
Precision_Conser_test	0.88	0.86	0.87	0.86

	LDA_test	Logistic_test	KNN_test	GaussianNb_test	SVM_test
Accuracy_test	0.82	0.82	0.80	0.83	0.82
AUC_test	0.88	0.88	0.86	0.88	0.89
Recall_Labour_test	0.88	0.89	0.85	0.87	0.88
Precision_Labour_test	0.87	0.87	0.87	0.89	0.87
Recall_Conser_test	0.67	0.65	0.68	0.72	0.65
Precision_Conser_test	0.88	0.89	0.85	0.87	0.88

- Here Model performs better in its default parameters.
- Boosting and Bagging Classifier performs too well on train data
- When it comes to test data,all of the models performs equally well
- Here both parametric and non parametric models performs well on the given data.
- Gaussian Naïve Bayes performs and Gradient Boost performs equally well on test set when compared to all other models.
- Gaussian NB and Gradient Boost is able to perform well on unknown data i.e it is able to generalize well for upcoming data points.
- For predicting further unknown data we could use either Gaussian NB model or Gradient Boost Classifier
- Since here the good performing models belong to both non parametric and parametric model, we are here to unable to confirm whether really the points comes from certain distribution.

CONCLUSION/INFERENCES

- People who want better national economic growth preferred Labour Party.
- People with different scales of 'political knowledge' always preferred Labour Party.
- Voters who are more Eurosceptic tend to vote Conservative Party.
- Data consists of more supporters of Labour Party, this could be due to two reasons , one would be bias in while sampling and the other would be in fact the Labour Party would really popular among the voters.
- Even though we were able to see imbalanced class scenario from the data points we didn't apply SMOTE to treat this issue.

- SMOTE produces artificial data which affects natural characteristics of the data, since data is regarding the exit-poll introducing artificial data would not make sense.
- All the models is at its highest performance when it is trained with default parameters, tuning didn't have any effect on the model.
- Since Gaussian Nb and Gradient Boosting performs well on test and train data we are unable to confirm whether data points comes from certain distribution.

THE END

NLP:
An Analysis on Speech
By
US Presidents

FINDING THE NO: OF WORDS,CHARACTERS AND SENTENCES:

- ROOSEVELT SPEECH:

```
The number of sentences =  68
The number of words =  1536
The number of characters =  6174
```

- KENNEDY SPEECH:

```
The number of sentences =  52
The number of words =  1546
The number of characters =  6202
```

- NIXON SPEECH:

```
The number of sentences =  69
The number of words =  2028
The number of characters =  8122
```

REMOVING STOPWORDS FROM DOCUMENT:

- NOTHING TO DESCRIBE HERE , FIND DETAILS IN PYTHON NOTEBOOK

FINDING MOST OCCURENCES OF WORDS(EXCLUDING STOPWORDS):

- ROOSEVELT'S SPEECH:

```
[('nation', 12), ('know', 10), ('spirit', 9)]
```

- KENNEDY'S SPEECH:

```
[('let', 16), ('us', 12), ('world', 8)]
```

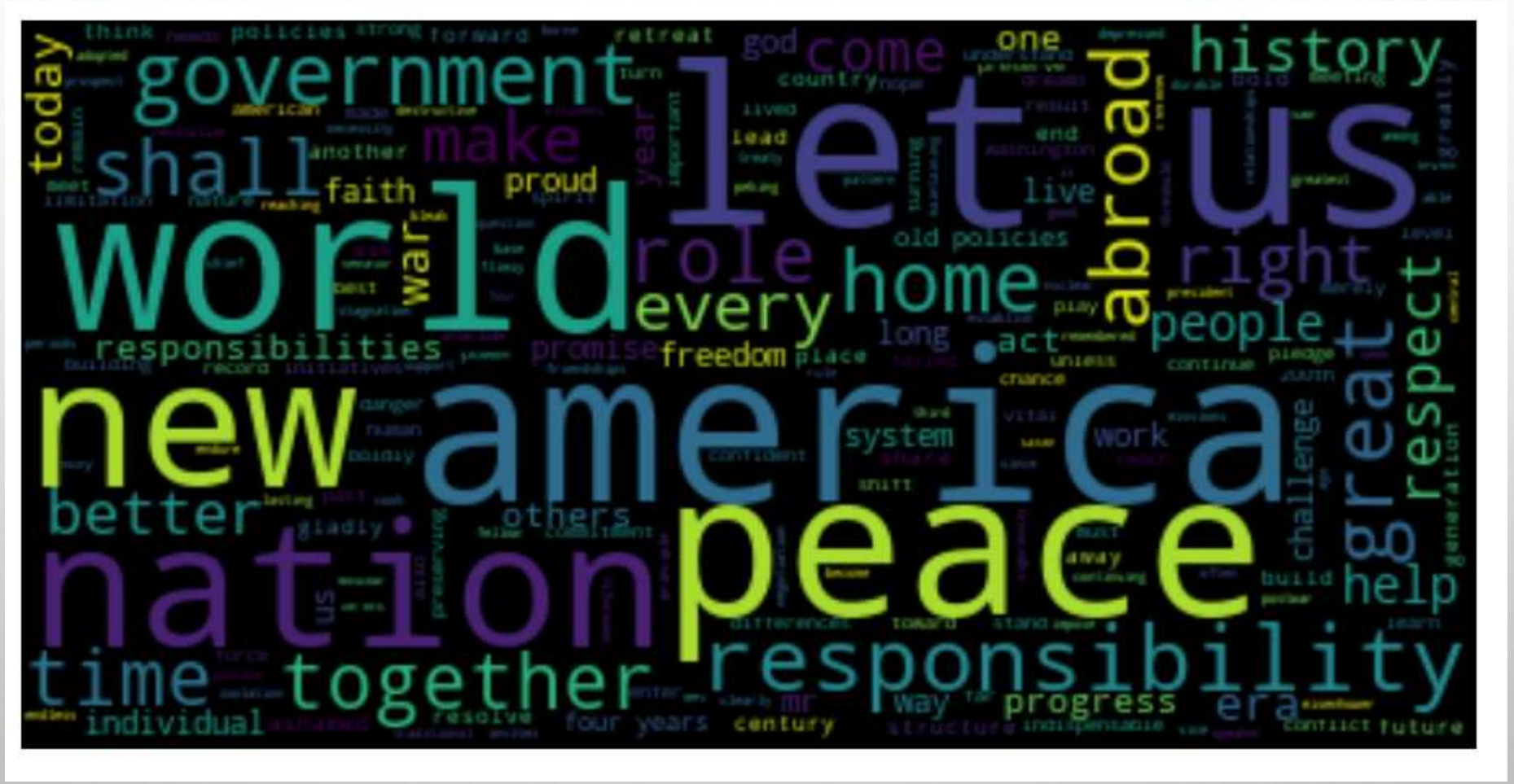
- NIXON'S SPEECH:

```
[('us', 26), ('let', 22), ('america', 21)]
```

WORD CLOUD ON ROOSEVELT'S SPEECH:

WORD CLOUD ON KENNEDY'S SPEECH:

WORD CLOUD ON NIXON'S SPEECH:



THE END