1.1 Read the dataset. Do the descriptive statistics and do null value condition check. Write an inference on it.

Reading the top 10 Records

Out[5]:

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge
0	Labour	43	3	3	4	1	2	2
1	Labour	36	4	4	4	4	5	2
2	Labour	35	4	4	5	2	3	2
3	Labour	24	4	2	2	1	4	0
4	Labour	41	2	2	1	1	6	2
5	Labour	47	3	4	4	4	4	2
6	Labour	57	2	2	4	4	11	2
7	Labour	77	3	4	4	1	1	0
8	Labour	39	3	3	4	4	11	0
9	Labour	70	3	2	5	1	11	2

Checking the datatype of the variables

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 9 columns):
vote                       1525 non-null object
age                        1525 non-null int64
economic.cond.national     1525 non-null int64
economic.cond.household    1525 non-null int64
Blair                      1525 non-null int64
Hague                      1525 non-null int64
Europe                     1525 non-null int64
political.knowledge        1525 non-null int64
gender                     1525 non-null object
dtypes: int64(7), object(2)
memory usage: 107.4+ KB
```

Converting political.knowledge Variable to Object from Integer

- Important Note:- political.knowledge variable having 0 values , if would be multipled by 0 , whole result would be 0

Checking the datatype after converting to political.knowledge to integer to object

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 9 columns):
vote                          1525 non-null object
age                           1525 non-null int64
economic.cond.national        1525 non-null int64
economic.cond.household       1525 non-null int64
Blair                         1525 non-null int64
Hague                         1525 non-null int64
Europe                        1525 non-null int64
political.knowledge           1525 non-null object
gender                        1525 non-null object
dtypes: int64(6), object(3)
memory usage: 107.4+ KB
```

Checking the data Description

Out[25]:

	age	economic.cond.national	economic.cond.household	Blair	Hague	Euro
count	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000	1525.0000
mean	54.182295	3.245902	3.140328	3.334426	2.746885	6.7285
std	15.711209	0.880969	0.929951	1.174824	1.230703	3.2975
min	24.000000	1.000000	1.000000	1.000000	1.000000	1.0000
25%	41.000000	3.000000	3.000000	2.000000	2.000000	4.0000
50%	53.000000	3.000000	3.000000	4.000000	2.000000	6.0000
75%	67.000000	4.000000	4.000000	4.000000	4.000000	10.0000
max	93.000000	5.000000	5.000000	5.000000	5.000000	11.0000

Checking the columns of the dataset

Out[9]:

```
Index(['vote', 'age', 'economic.cond.national', 'economic.cond.household',
       'Blair', 'Hague', 'Europe', 'political.knowledge', 'gender'],
      dtype='object')
```

Checking the shape of data

Out[10]:

```
(1525, 9)
```

Checking the dataset missing values?

Out[11]:

	Total	Percent
gender	0	0.0
political.knowledge	0	0.0
Europe	0	0.0
Hague	0	0.0
Blair	0	0.0
economic.cond.household	0	0.0
economic.cond.national	0	0.0
age	0	0.0
vote	0	0.0

Checking the Duplicates Values

Out[12]: 8

Removing the duplicate Values

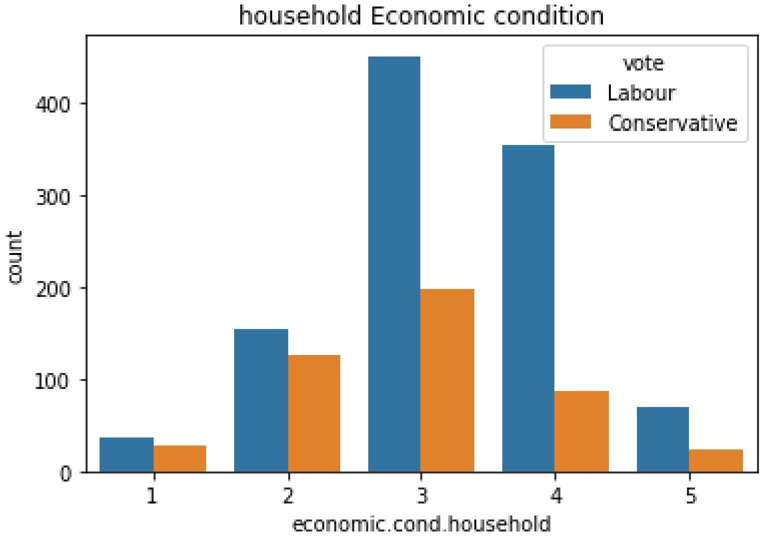
Cross checking the duplicate values after Removal

Out[14]: 0

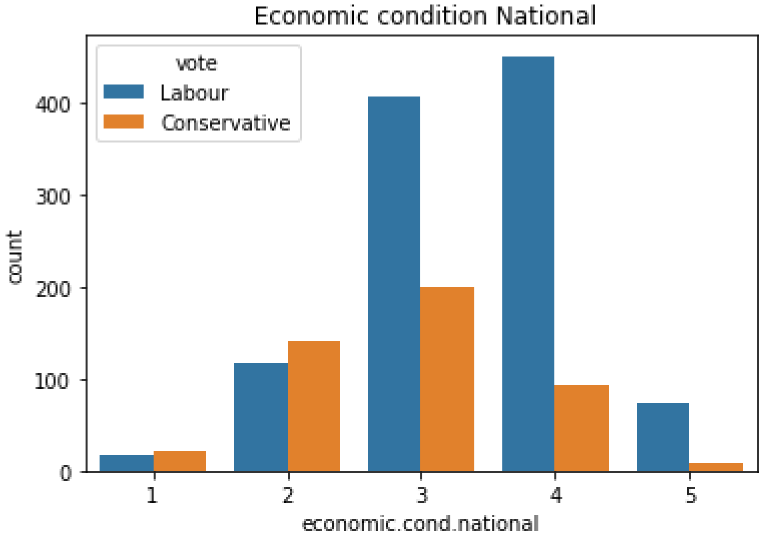
```
VOTE :  2
Conservative     462
Labour          1063
Name: vote, dtype: int64


GENDER :  2
male      713
female    812
Name: gender, dtype: int64
```

Count plot between vote and Household Economic Condition

Out[332]: Text(0.5, 1.0, 'household Economic condition')

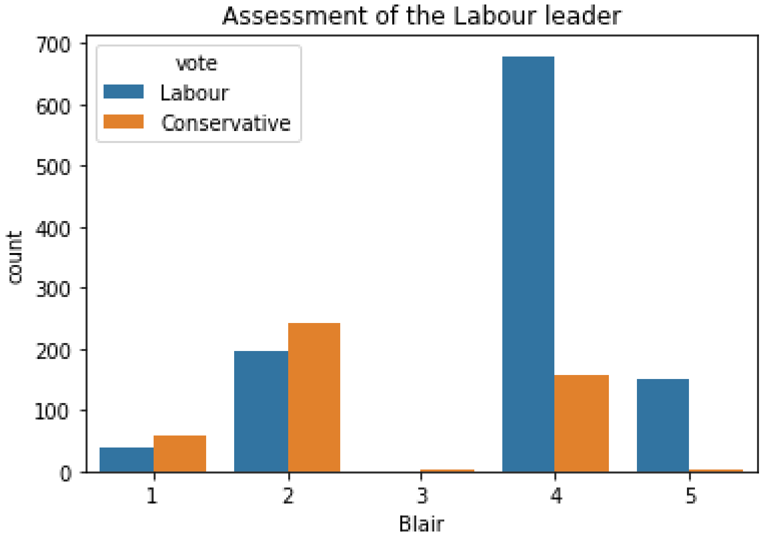


Out[334]: Text(0.5, 1.0, ' Economic condition National')



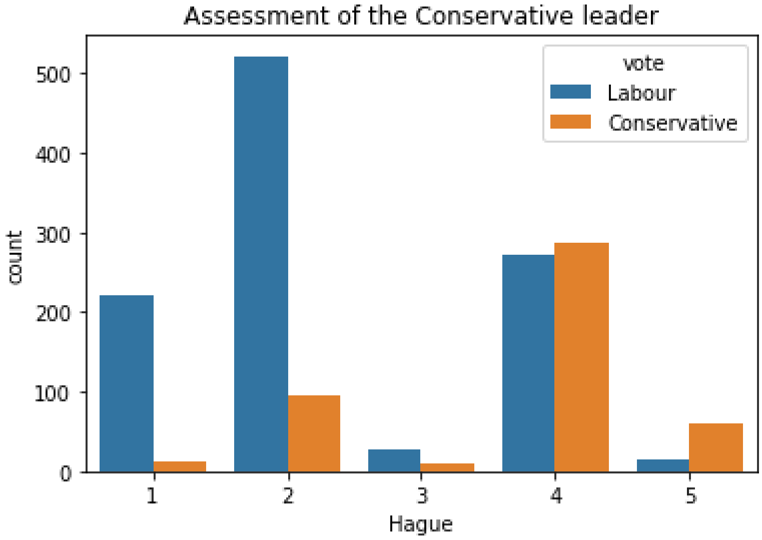
Count plot between vote and Assessment of the Labour leader

Out[333]: Text(0.5, 1.0, 'Assessment of the Labour leader')



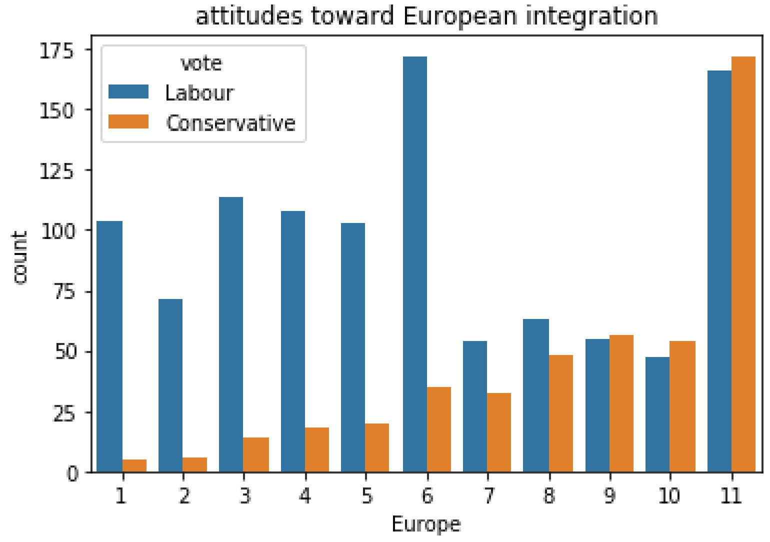
Count plot between vote and Assessment of the Conservative leader

Out[18]: Text(0.5, 1.0, 'Assessment of the Conservative leader')

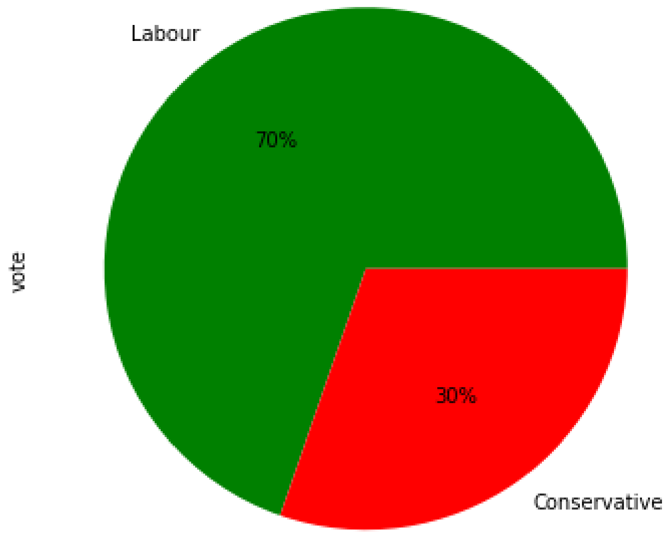


Count plot between vote and Attitudes toward European integration

Out[19]: Text(0.5, 1.0, 'attitudes toward European integration')

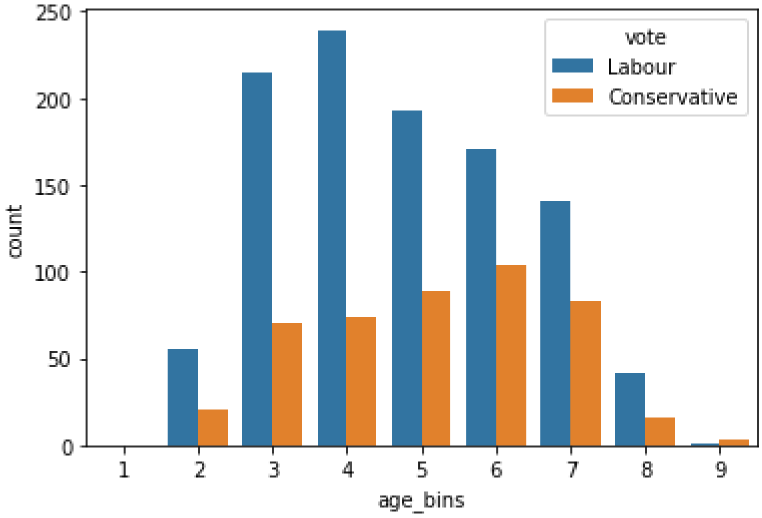


Pie Chart of Vote distribution



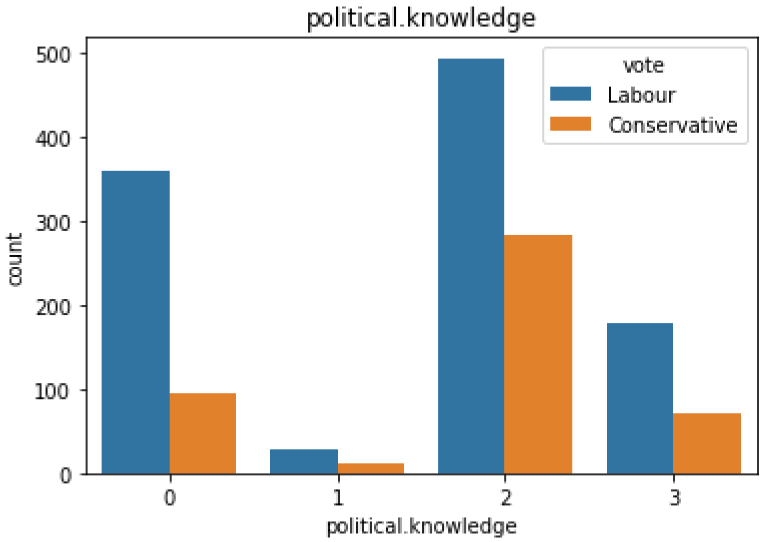
Count plot between Vote and Age

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x2bbfb4fff88>

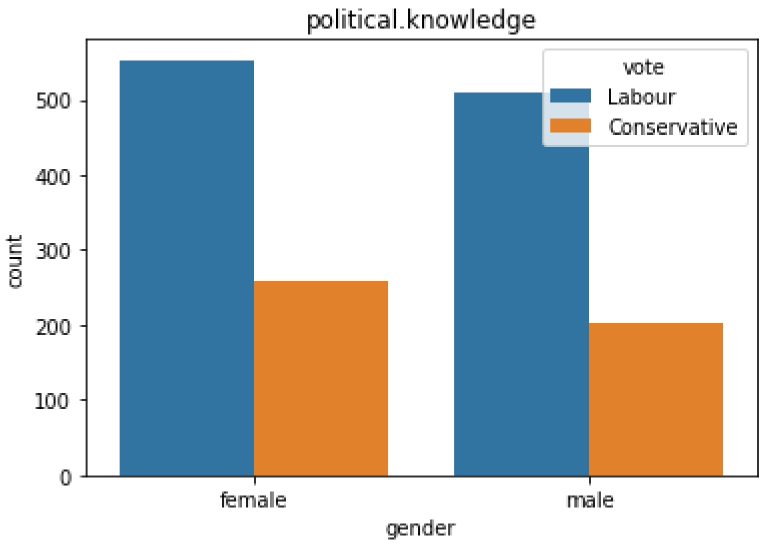


Count plot between vote and political.knowledge

Out[23]: Text(0.5, 1.0, 'political.knowledge')



Out[7]: Text(0.5, 1.0, 'political.knowledge')

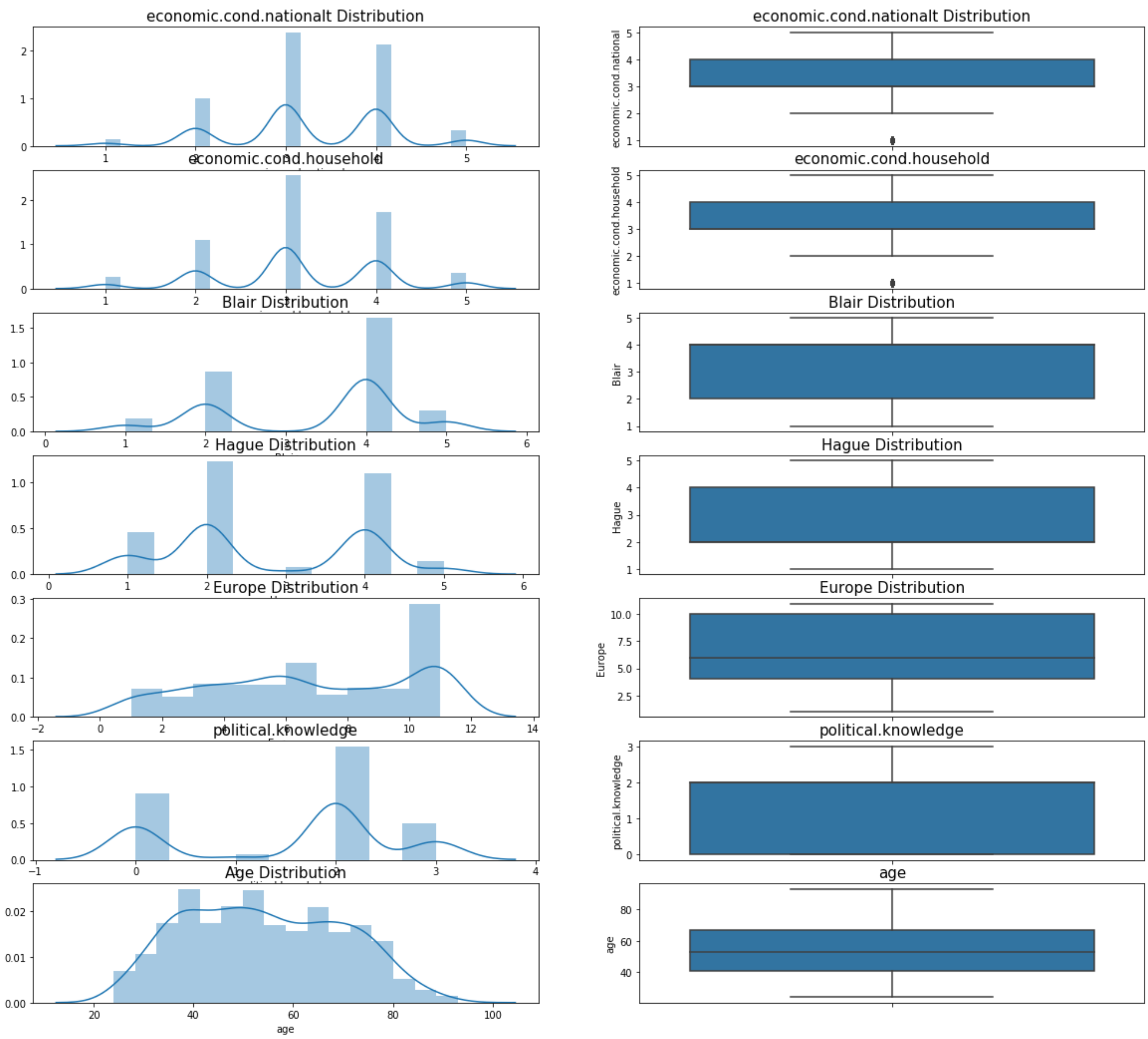


1.1) Read the dataset. Do the descriptive statistics and do null value condition check.

- There is no null values in the dataset .
- There are duplicates values has been observed in the dataset .
- Important Note :- Data of conservative and Labour candidate is 30 and 70 percent.(it is imbalanced dataset)
- There 1525 rows 9 columns in the given dataset
- Vote is the target Variable
- There are age,economic.cond.national,economic.cond.household,Blair,Hague,Europe,political.knowledge and gender are independent variables in the dataset

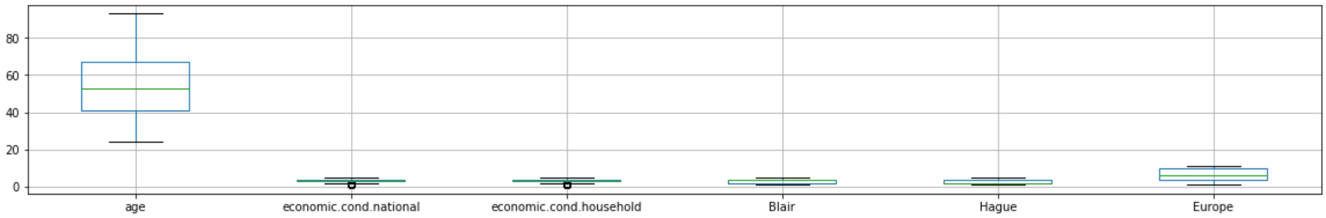
1.2) Perform Univariate and Bivariate Analysis. Do exploratory data analysis. Check for Outliers.

Perform Univariate Analysis

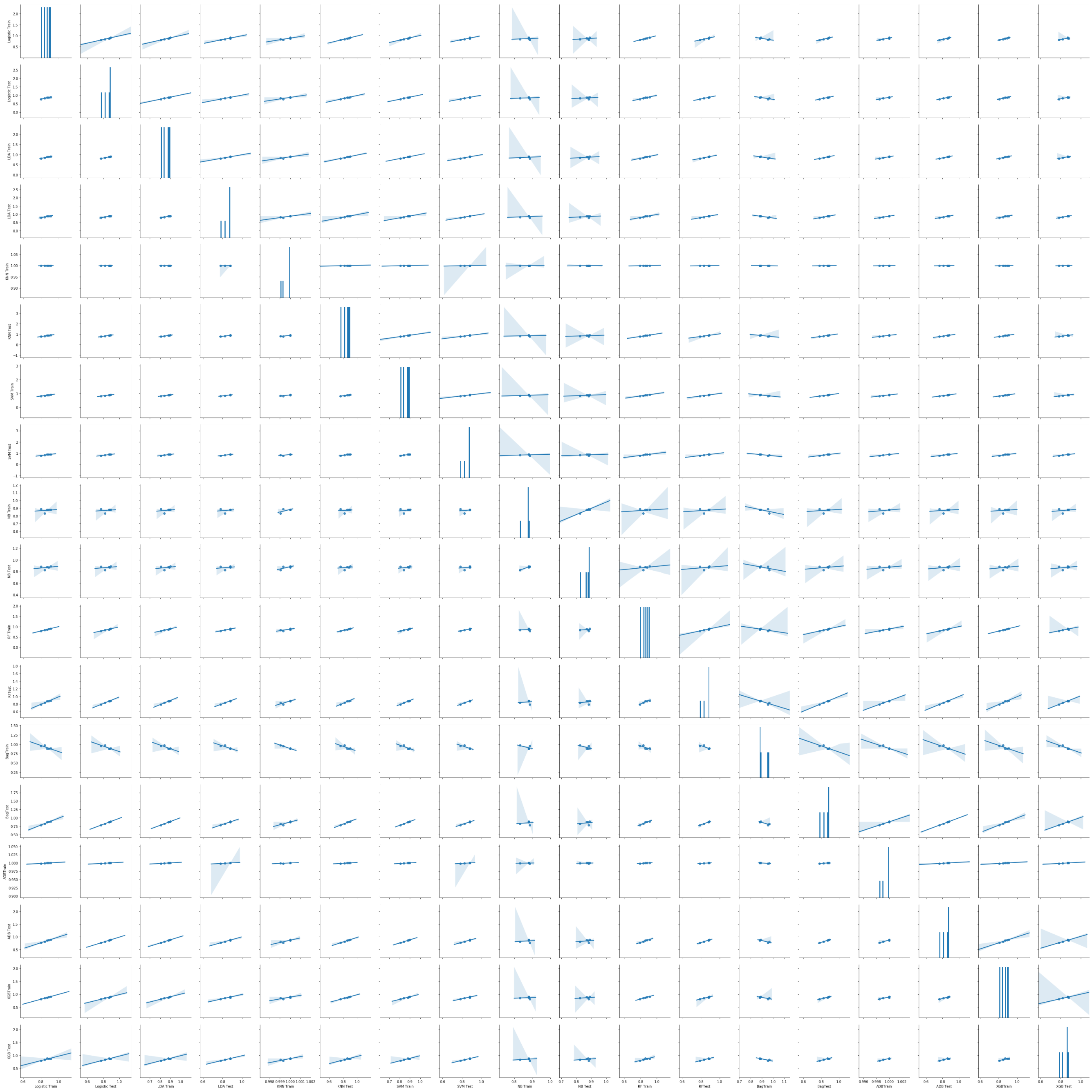


Checking the outliers

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x2bbfb65db08>



Perform Bivariate Analysis

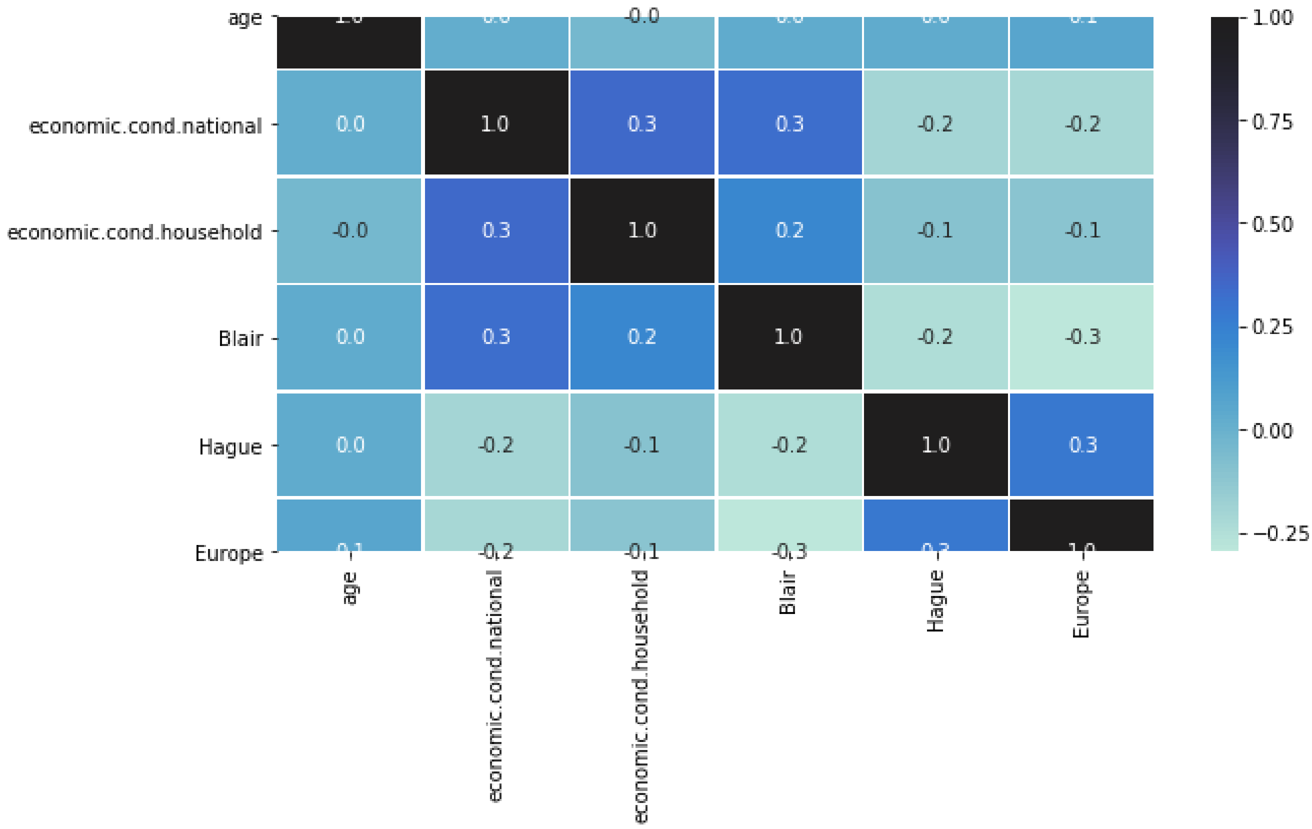


Correlation among pairs of continuous variables

Out[26]:

	age	economic.cond.national	economic.cond.household	Blair	Hague
age	1.000000	0.018687	-0.038868	0.032084	0.031144
economic.cond.national	0.018687	1.000000	0.347687	0.326141	-0.200790
economic.cond.household	-0.038868	0.347687	1.000000	0.215822	-0.100392
Blair	0.032084	0.326141	0.215822	1.000000	-0.243508
Hague	0.031144	-0.200790	-0.100392	-0.243508	1.000000
Europe	0.064562	-0.209150	-0.112897	-0.295944	0.285738

Heatmap among continuous variables



1.2) Perform Univariate and Bivariate Analysis. Do exploratory data analysis. Check for Outliers.

- It depend the outliers defination what we define , whether particular dataaset having outliers or not . If we take the defination of outliers that is as below
- lower range = Q1-1.5*IQR
- upper range=Q1+1.5*IQR
- If we consider above defination , yes outliers exist in variable like economic.cond.national and economic.cond.household
- But These outlieres are very important for doing the analysis . These outliers can not be removed .
- Variable National Econnomic and House hold economic ondition are equally distributted
- Political knowlege,age ,Blair and Hague variable are not normally distributed
- National economic condition and Blair are 30 percent postively corelated
- National economic condition and Household economic condition are 30 percent postively corelated
- National economic condition and Hauge are slighty negatively corelated(.20)
- National economic condition and Europe are slighty negatively corelated(.20)
- Hauge,Blair ,Europe variable play important role in decision making

Data Preparation:1.3 Encode the data (having string values) for Modelling. Is Scaling necessary here or not? Data Split: Split the data into train and test (70:30).

Checking the dataset information

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1517 entries, 0 to 1524
Data columns (total 10 columns):
vote                       1517 non-null object
age                        1517 non-null int64
economic.cond.national     1517 non-null int64
economic.cond.household    1517 non-null int64
Blair                      1517 non-null int64
Hague                      1517 non-null int64
Europe                     1517 non-null int64
political.knowledge        1517 non-null object
gender                     1517 non-null object
age_bins                   1517 non-null category
dtypes: category(1), int64(6), object(3)
memory usage: 160.4+ KB
```

Removal of the age_bins column from the dataset

After removal of the age_bins column from the dataset crosscheck dataset info

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 9 columns):
vote                       1525 non-null int8
age                        1525 non-null int64
economic.cond.national     1525 non-null int64
economic.cond.household    1525 non-null int64
Blair                      1525 non-null int64
Hague                      1525 non-null int64
Europe                     1525 non-null int64
political.knowledge        1525 non-null int64
gender                     1525 non-null int8
dtypes: int64(7), int8(2)
memory usage: 86.5 KB
```

Data Preparation:Checking the proportions of the columns values

```
VOTE :  2
Conservative     460
Labour          1057
Name: vote, dtype: int64


POLITICAL.KNOWLEDGE :  4
1     38
3    249
0    454
2    776
Name: political.knowledge, dtype: int64


GENDER :  2
male      709
female    808
Name: gender, dtype: int64
```

Data Preparation:1.3 Encode the data (having string values) for Modelling

```
feature: vote
[Labour, Conservative]
Categories (2, object): [Conservative, Labour]
[1 0]


feature: gender
[female, male]
Categories (2, object): [female, male]
[0 1]
```

Train Test Split

Split X and y into training and test set in 70:30 ratio

1.3) Is Scaling necessary here or not?

- if We See the data , Most of the column having data pertaining to rating or people preference (people does not like the candidate the give the rating 0 or 1 and prople like the candiate give the rating 4,5) . In such kind of data, scaling is not required

1.4) Apply Logistic Regression and LDA (Linear Discriminant Analysis).

Logistic Regression

Out[92]:
```
LogisticRegression()
```

Basic Model Score (Logistic Regression)

```
0.8397375820056232
0.8231441048034934
```

HyperTuning parameter (Logistic Regression)

```
{'C': 1.0, 'penalty': 'l1', 'solver': 'liblinear'}
```

Out[95]:
```
LogisticRegression(penalty='l1', solver='liblinear')
```

Predicting on Training and Test dataset(Logistic Regression)

Performance Evaluation on Training data(Logical Regression)

Confusion Matrix of Training Data(Logical Regression)

Out[97]:
```
array([[231, 101],
       [ 68, 667]], dtype=int64)
```

Accuracy Score of Training Data(Logical Regression)

Out[98]:
```
0.8416119962511716
```

Classification Report of Training Data(Logical Regression)

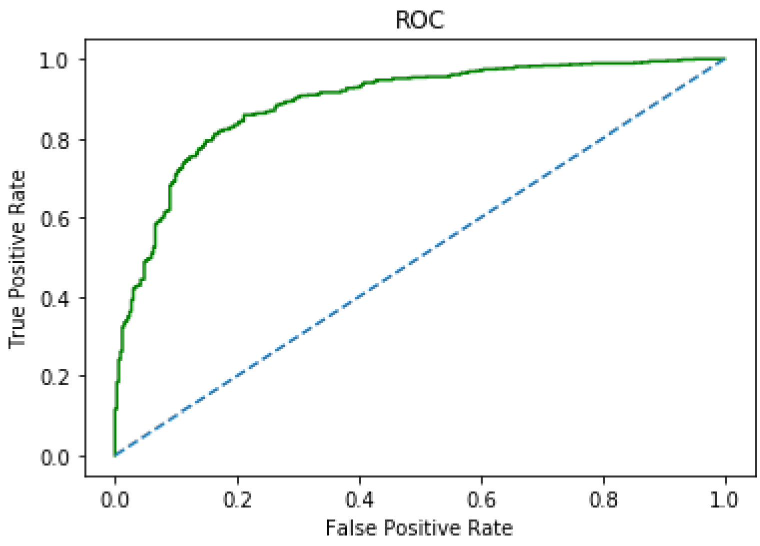
```
              precision    recall  f1-score   support

           0       0.77      0.70      0.73       332
           1       0.87      0.91      0.89       735

    accuracy                           0.84      1067
   macro avg       0.82      0.80      0.81      1067
weighted avg       0.84      0.84      0.84      1067
```

Training data Metrics(Logistic Regression)

```
Logical_train_precision  0.91
Logical_train_recall  0.89
Logical_train_f1  0.87
```

AUC and ROC of Training Data(Logical Regression)

Area under Curve is 0.8016330628636996



Performance Evaluation on Test data(Logical Regression)

Confusion Matrix Test Data(Logical Regression)

Out[102]:
```
array([[ 85,  45],
       [ 37, 291]], dtype=int64)
```

Accuracy Score of Test Data(Logical Regression)

Out[103]:
```
0.8209606986899564
```

Confusion Matrix for test data((Logistic Regression)

Out[104]:
```
array([[ 85,  45],
       [ 37, 291]], dtype=int64)
```

Test data Accuracy (Logistic Regression)

Out[105]:
```
0.8209606986899564
```

Classification Report of Test data (Logistic Regression)

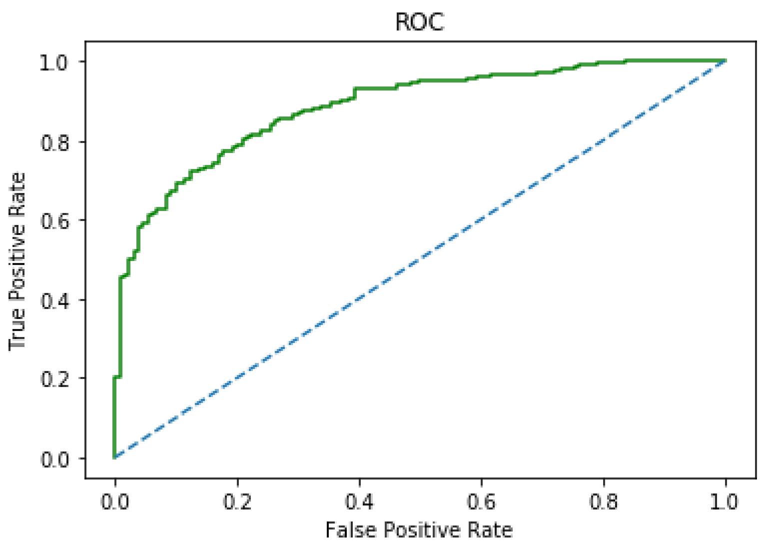
```
              precision    recall  f1-score   support

           0       0.70      0.65      0.67       130
           1       0.87      0.89      0.88       328

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

Test Metrics (Logistic Regression)

```
Logical_test_precision  0.89
Logical_test_recall  0.88
Logical_test_f1  0.87
```

AUC and ROC of Test Data(Logical Regression)

```
Area under Curve is 0.7705206378986866
```



Train and Test Performance (Logistic Regression)

Out[89]:

	LogisticTrain	Logistic Test
Accuracy	0.84	0.82
AUC	0.80	0.77
Recall	0.89	0.88
Precision	0.91	0.89
F1 Score	0.87	0.87

Logistic Regression Model Final comments

- Basic Logistic Regression Model having training and test score is 83.9% and 82.3 %
- Tuning Logistic Regression Model having training and test score is 84% and 82 %
- Both Basic and Tuning Model both are giving similar results and They are Right fit Model ie Neither overfit and Not underfit

LDA Model

Out[143]:
```
LinearDiscriminantAnalysis()
```

Basic Model Score (LDA)

```
0.7835051546391752
0.7860262008733624
```

HyperTuning parameter (LDA)

```
{'shrinkage': 'auto', 'solver': 'lsqr'}
```

Out[145]:
```
LinearDiscriminantAnalysis(shrinkage='auto', solver='lsqr')
```

Predicting the Training and Testing data(LDA)

Performance Evaluation on Training data(LDA)

Confusion Matrix of Training Data(LDA)

Out[147]:
```
array([[238,  94],
       [ 76, 659]], dtype=int64)
```

Accuracy Score of Training Data(LDA)

Out[148]:
```
0.8406747891283973
```

Classification Report of Training Data(LDA)

```
              precision    recall  f1-score   support

           0       0.76      0.72      0.74       332
           1       0.88      0.90      0.89       735

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

Training Matrics (LDA)

```
LDA_train_precision  0.9
LDA_train_recall  0.89
LDA_train_f1  0.88
```

AUC and ROC of Training Data(LDA)

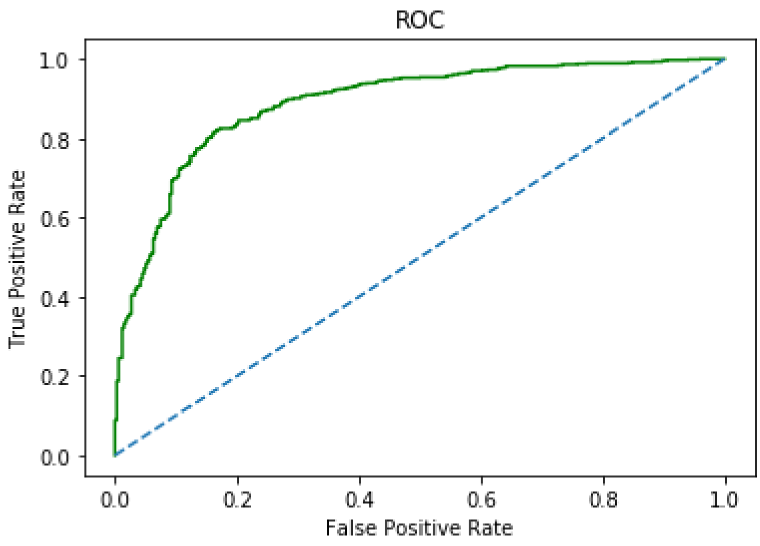
```
              precision    recall  f1-score   support

           0       0.74      0.65      0.69       307
           1       0.86      0.91      0.89       754

    accuracy                           0.83      1061
   macro avg       0.80      0.78      0.79      1061
weighted avg       0.83      0.83      0.83      1061
```

LDA training matrics

```
LDA_train_precision  0.86
LDA_train_recall  0.91
LDA_train_f1  0.89
```

AUC and ROC of Training Data(LDA)

```
Area under Curve is 0.8067330546676502
```



Performance Evaluation on Test data(LDA)

Confusion Matrix Test Data(LDA)

Out[152]:
```
array([[ 89,  41],
       [ 40, 288]], dtype=int64)
```

Accuracy Score of Test Data(LDA)

Out[153]:
```
0.8231441048034934
```

Classification Report of Test Data(LDA)

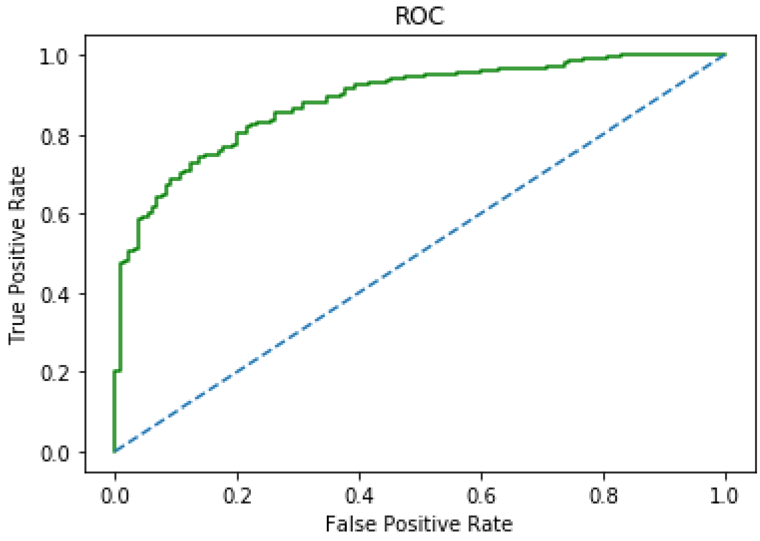
```
              precision    recall  f1-score   support

           0       0.69      0.68      0.69       130
           1       0.88      0.88      0.88       328

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

Test Data Matrics(LDA)

```
LDA_test_precision  0.88
LDA_test_recall  0.88
LDA_test_f1  0.88
```

AUC and ROC of Test Data(LDA)

```
Area under Curve is 0.7813320825515948
```



Train and Test Performance (LDA)

Out[161]:

	LDATrain	LDA Test
Accuracy	0.84	0.82
AUC	0.81	0.78
Recall	0.89	0.88
Precision	0.90	0.88
F1 Score	0.88	0.88

LDA Model Final comments

- Basic LDA Model having training and test score is 83.6% and 81.8 %
- Tuning LDA Model having training and test score is 84% and 82 %
- Both Basic and Tuning Model both are giving similar results and Right fit Model ie Neither overfit and Not underfit

1.5) Apply KNN Model, Naïve Bayes Model and Support Vector Machine (SVM) model.

KNN Model

Out[29]:
```
KNeighborsClassifier()
```

Basic Model Score (KNN model)

```
Training 0.8537956888472352
Testing 0.7860262008733624
```

HyperTuning parameter (KNN)

Predicting the Training and Testing data(KNN)

```
{'metric': 'manhattan', 'n_neighbors': 41, 'weights': 'distance'}
```

Out[23]:
```
KNeighborsClassifier(metric='manhattan', n_neighbors=41, weights='distance')
```

Predicting the Training and Testing data(KNN)

Performance Evaluation on Training data(KNN)

Confusion Matrix of Training Data(KNN)

Out[181]:
```
array([[332,   0],
       [  1, 734]], dtype=int64)
```

Accuracy Score of Training Data(KNN)

Out[25]:
```
0.9990627928772259
```

Classification Report of Training Data(KNN)

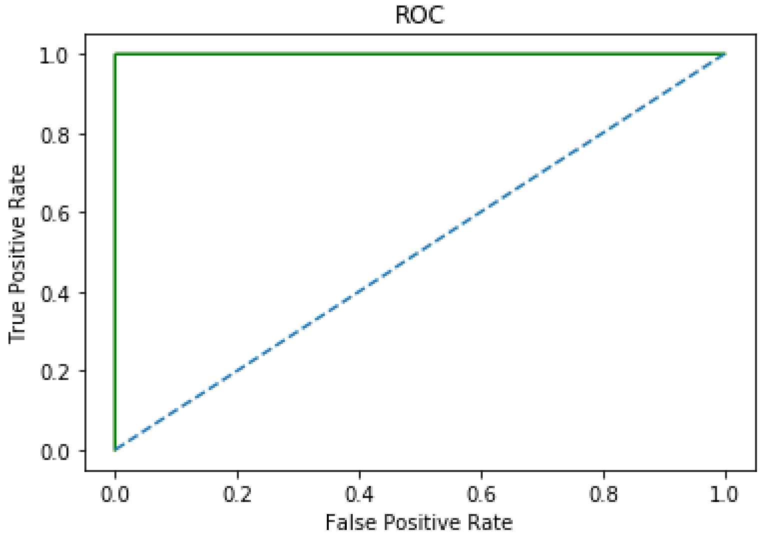
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       332
           1       1.00      1.00      1.00       735

    accuracy                           1.00      1067
   macro avg       1.00      1.00      1.00      1067
weighted avg       1.00      1.00      1.00      1067
```

Training Matrics (KNN)

```
KNN_train_precision  1.0
KNN_train_recall  1.0
KNN_train_f1  1.0
```

AUC and ROC of Training Data(KNN)

```
Area under Curve is 0.9993197278911565
```



Performance Evaluation on Test data(KNN)

Confusion Matrix Test Data(KNN)

Out[186]:
```
array([[ 81,  49],
       [ 33, 295]], dtype=int64)
```

Accuracy Score of Test Data(KNN)

Out[26]:
```
0.8209606986899564
```

Classification Report of Test Data(KNN)

	precision	recall	f1-score	support
0	0.71	0.62	0.66	130
1	0.86	0.90	0.88	328
accuracy			0.82	458
macro avg	0.78	0.76	0.77	458
weighted avg	0.82	0.82	0.82	458

Test Data Accuracy(KNN)

Out[76]:
```
0.8157894736842105
```

Classification Report of Test Data (KNN)

```
              precision    recall  f1-score   support

           0       0.77      0.65      0.70       153
           1       0.83      0.90      0.87       303

    accuracy                           0.82       456
   macro avg       0.80      0.77      0.78       456
weighted avg       0.81      0.82      0.81       456
```

KNN Test matrics

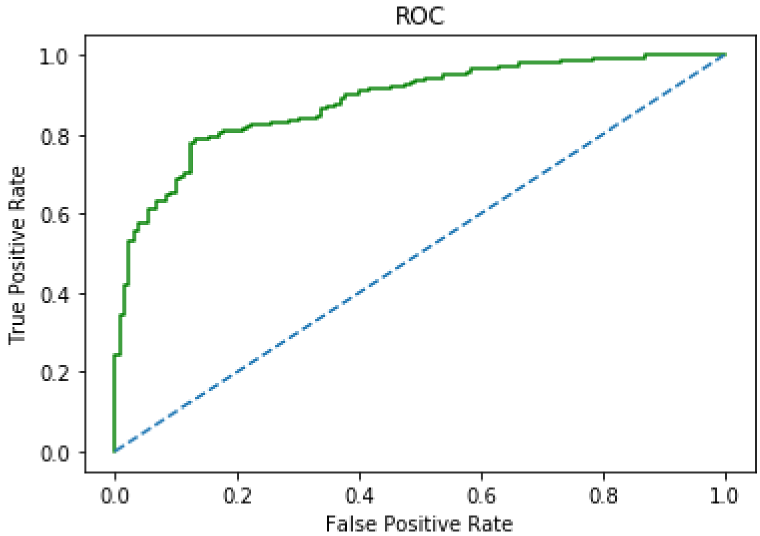
```
KNN_test_precision  0.83
KNN_test_recall  0.9
KNN_test_f1  0.87
```

Test Data Matrics(KNN)

```
KNN_test_precision  0.9
KNN_test_recall  0.88
KNN_test_f1  0.86
```

AUC and ROC of Test Data(KNN)

```
Area under Curve is 0.7612335834896811
```



Train and Test Performance (KNN)

Out[193]:

	KNN Train	KNN Test
Accuracy	1.0	0.82
AUC	1.0	0.76
Recall	1.0	0.88
Precision	1.0	0.90
F1 Score	1.0	0.86

KNN Model Final comments

- Basic KNN Model having training and test score is 85.3 % and 78.6 % respectively
- Tuning KNN Model having training and test score is 99% and 82 % respectively
- Basic Model is Right fit Model While Tunning Model is overfitting

SVM Model

Out[35]:
```
SVC(probability=True)
```

Basic Model Score (SVM model)

```
Training 0.7835051546391752
Testing 0.7860262008733624
```

HyperTuning parameter (SVM)

```
{'C': 10, 'gamma': 0.001, 'kernel': 'rbf'}
```

Out[138]:
```
SVC(C=10, gamma=0.001)
```

Predicting the Training and Testing data(SVM)

Performance Evaluation on Training data(SVM)

Confusion Matrix of Training Data(SVM)

Out[163]:
```
array([[238,  94],
       [ 76, 659]], dtype=int64)
```

Accuracy Score of Training Data(SVM)

Out[164]:
```
0.8406747891283973
```

Classification Report of Training Data(SVM)

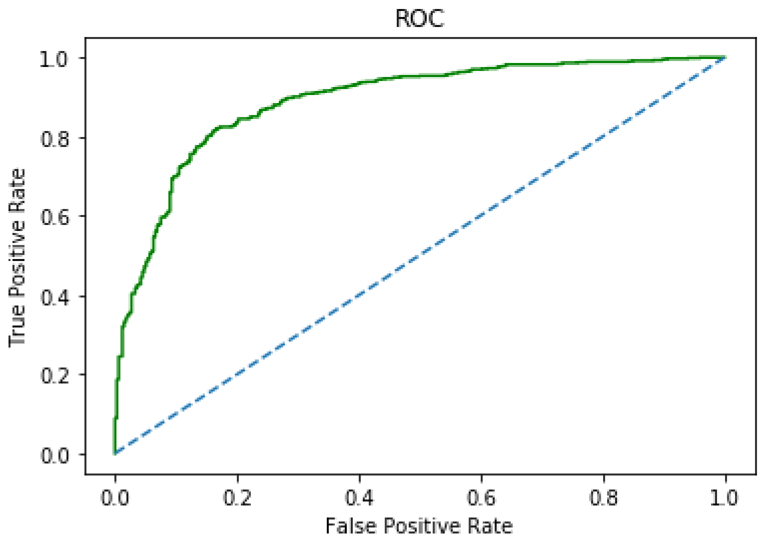
```
              precision    recall  f1-score   support

           0       0.76      0.72      0.74       332
           1       0.88      0.90      0.89       735

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

Training Matrics (SVM)

```
SVM_train_precision  0.9
SVM_train_recall  0.89
SVM_train_f1  0.88
```

AUC and ROC of Training Data(SVM)

```
Area under Curve is 0.8067330546676502
```



Confusion Matrix for the training data(SVM)

```
Out[90]: array([[ 26, 281],
                [  3, 751]], dtype=int64)
```

Classification Report of training data (SVM)

	precision	recall	f1-score	support
0	0.90	0.08	0.15	307
1	0.73	1.00	0.84	754
accuracy			0.73	1061
macro avg	0.81	0.54	0.50	1061
weighted avg	0.78	0.73	0.64	1061

SVM training matrics

```
SVM_train_precision  0.73
SVM_train_recall  1.0
SVM_train_f1  0.84
```

Performance Evaluation on Test data(SVM)

Confusion Matrix Test Data(SVM)

```
Out[168]: array([[ 89,  41],
                 [ 40, 288]], dtype=int64)
```

Accuracy Score of Test Data(SVM)

Out[169]: 0.8231441048034934

Classification Report of Test Data(SVM)

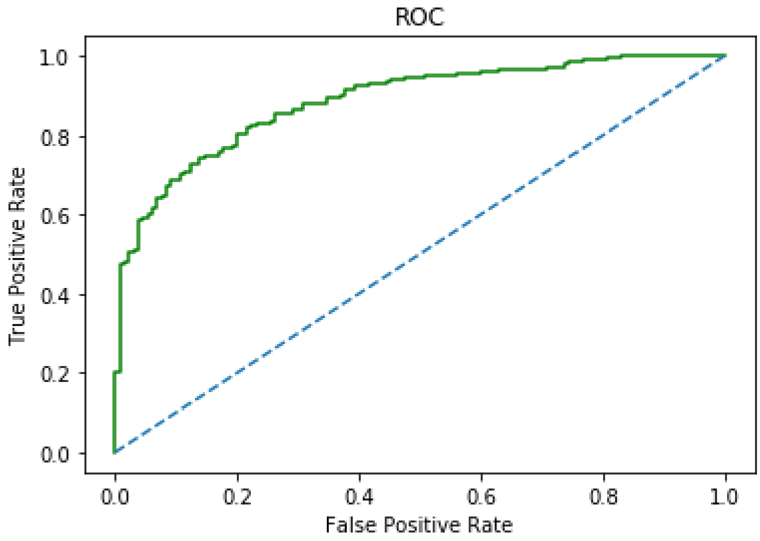
```
              precision    recall  f1-score   support

           0       0.69      0.68      0.69       130
           1       0.88      0.88      0.88       328

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

Test Data Matrics(SVM)

```
SVM_test_precision  0.88
SVM_test_recall  0.88
SVM_test_f1  0.88
```

AUC and ROC of Test Data(SVM)

Area under Curve is 0.7813320825515948



Train and Test Performance (SVM)

Out[174]:

	SVMTrain	SVM Test
Accuracy	0.84	0.82
AUC	0.81	0.78
Recall	0.89	0.88
Precision	0.90	0.88
F1 Score	0.88	0.88

SVM Model Final comments

- Basic SVM Model having training and test score is 78.3% and 78.6% respectively .
- Tuning SVM Model having training and test score is 84% and 82 % respectively .
- Basic and Tuning Model is Right fit .

Naive Bayes

Out[40]:
```
GaussianNB()
```

Basic Model Score (Naive Bayes)

```
Training 0.8331771321462043
Testing 0.8253275109170306
```

Predicting on Training and Test dataset(NB)

Getting the Predicted Classes and Probs(NB)

Out[293]:

	0	1
0	0.992582	0.007418
1	0.872464	0.127536
2	0.434483	0.565517
3	0.536044	0.463956
4	0.242177	0.757823

NB Model Evaluation

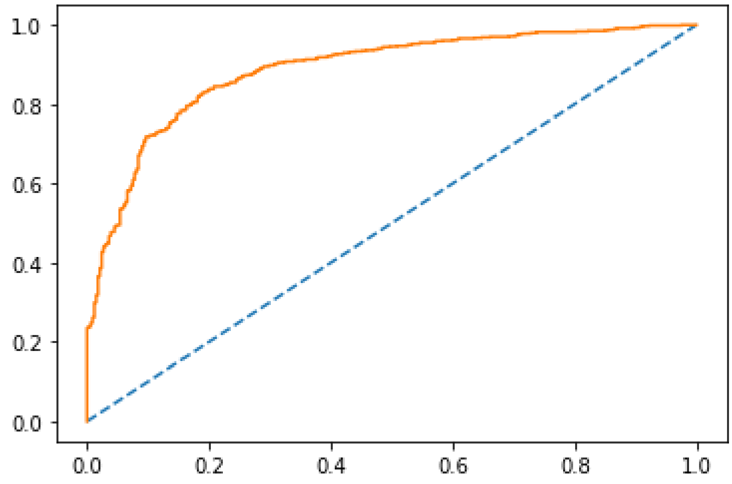
Training Data Accuracy (NB)

Out[294]:
```
0.8331771321462043
```

AUC and ROC for the training data(NB)

AUC: 0.886

Out[295]: [<matplotlib.lines.Line2D at 0x2e0fdabc7c8>]



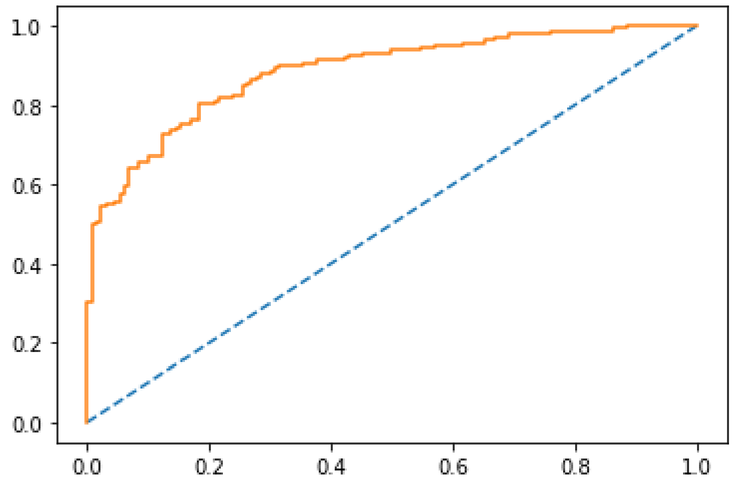
Test Data Accuracy (NB)

Out[296]: 0.8253275109170306

AUC and ROC for the test data (NB)

AUC: 0.885

Out[297]: [<matplotlib.lines.Line2D at 0x2e0fdaeee08>]



Confusion Matrix for the training data(NB)

Out[298]:
```
array([[240,  92],
       [ 86, 649]], dtype=int64)
```

Classification Report of training data (NB)

```
              precision    recall  f1-score   support

           0       0.74      0.72      0.73       332
           1       0.88      0.88      0.88       735

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

NB training matrics

```
NB_train_precision  0.88
NB_train_recall  0.88
NB_train_f1  0.88
```

Confusion Matrix for Test Data(NB)

Out[301]:
```
array([[ 94,  36],
       [ 44, 284]], dtype=int64)
```

Test Data Accuracy(NB)

Out[302]:
```
0.8253275109170306
```

Classification Report of Test Data (NB)

```
              precision    recall  f1-score   support

           0       0.68      0.72      0.70       130
           1       0.89      0.87      0.88       328

    accuracy                           0.83       458
   macro avg       0.78      0.79      0.79       458
weighted avg       0.83      0.83      0.83       458
```

NB Test matrics

```
NB_test_precision  0.89
NB_test_recall  0.87
NB_test_f1  0.88
```

Train and Test Performance (NB)

Out[305]:

	NB Train	NB Test
Accuracy	0.83	0.83
AUC	0.89	0.88
Recall	0.88	0.87
Precision	0.88	0.89
F1 Score	0.88	0.88

Naive Bayes Model Final comments

- Basic Naive Bayes Model having training and test score is 83.3% and 82.5% respectively .
- Basic Model is Right fit .

1.6) Model Tuning, Bagging and Boosting.

Building the Random Forest Base Model

Out[45]:
```
RandomForestClassifier()
```

Basic Model Score (Random Forest)

```
Training 0.9990627928772259
Testing 0.8187772925764192
```

HyperTuning parameter (Random Forest)

```
{'max_depth': 75, 'max_features': 4, 'min_samples_leaf': 25, 'min_samples_split': 50,
'n_estimators': 5}
```

Out[197]:
```
RandomForestClassifier(max_depth=75, max_features=4, min_samples_leaf=25,
                       min_samples_split=50, n_estimators=5)
```

Predicting the Training and Testing data(RF)

Performance Evaluation on Training data(RF)

Confusion Matrix of Training Data(RF)

Out[199]:
```
array([[225, 107],
       [ 69, 666]], dtype=int64)
```

Accuracy Score of Training Data(RF)

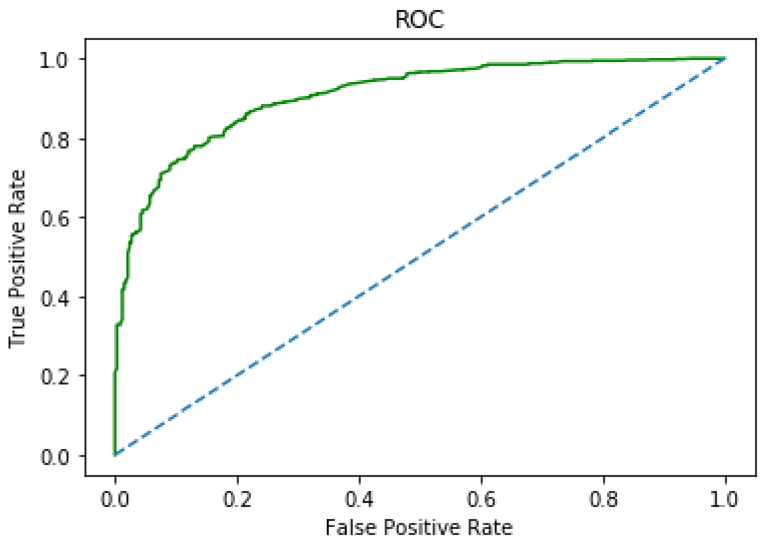
Out[200]:
```
0.8350515463917526
```

Classification Report of Training Data(RF)

```
              precision    recall  f1-score   support

           0       0.77      0.68      0.72       332
           1       0.86      0.91      0.88       735

    accuracy                           0.84      1067
   macro avg       0.81      0.79      0.80      1067
weighted avg       0.83      0.84      0.83      1067


rf_train_precision  0.91
rf_train_recall  0.88
rf_train_f1  0.86
```

AUC and ROC of Training Data(RF MODEL)

```
Area under Curve is 0.7919166461765429
```



Performance Evaluation on Test data(RF Model)

```
              precision    recall  f1-score   support

           0       0.71      0.68      0.70       130
           1       0.88      0.89      0.88       328

    accuracy                           0.83       458
   macro avg       0.79      0.79      0.79       458
weighted avg       0.83      0.83      0.83       458
```

Confusion Matrix Test Data(RF Model)

```
Out[205]: array([[ 89,  41],
                 [ 36, 292]], dtype=int64)
```

Accuracy Score of Test Data(RF)

Out[206]: 0.8318777292576419

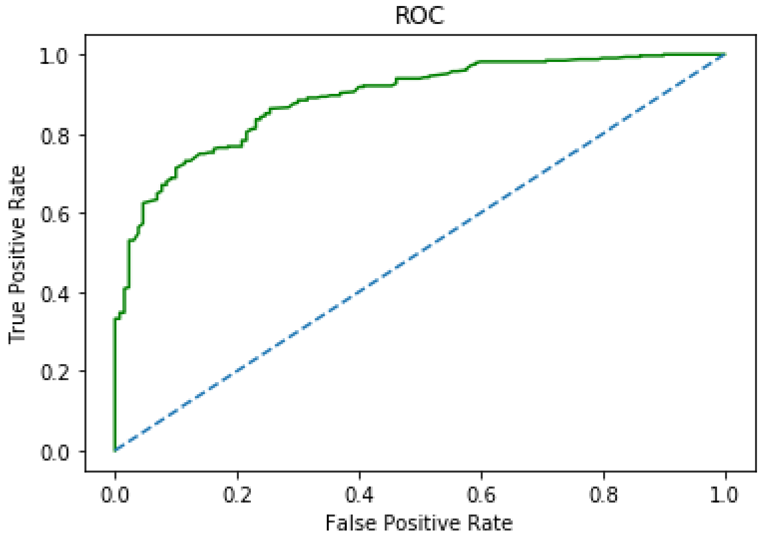
Classification Report of Test Data(RF Model)

	precision	recall	f1-score	support
0	0.71	0.68	0.70	130
1	0.88	0.89	0.88	328
accuracy			0.83	458
macro avg	0.79	0.79	0.79	458
weighted avg	0.83	0.83	0.83	458

```
rf_test_precision  0.89
rf_test_recall  0.88
rf_test_f1  0.88
```

AUC and ROC for the Test data(RF MODEL)

```
Area under Curve is 0.7874296435272046
```



Variable Importance(RF Model)

	Imp
Hague	0.333366
Europe	0.298936
Blair	0.196656
political.knowledge	0.070178
age	0.055860
economic.cond.national	0.028031
economic.cond.household	0.010895
gender	0.006077

Out[212]:

	Random Forest Train	Random Forest Test
Accuracy	0.84	0.83
AUC	0.79	0.79
Recall	0.88	0.88
Precision	0.91	0.89
F1 Score	0.86	0.88

Random Forest Model Final comments

- Basic Random Forest Model having training and test score is 99.9% and 81.7% respectively .
- Tuning Random Forest Model having training and test score is 83.5% and 83.1% respectively .
- Basic Model is overfitting while Tuning Model is Right fit .

Bagging

Out[214]:
```
BaggingClassifier(base_estimator=RandomForestClassifier())
```

Basic Model Score (Bagging Model)

```
0.9625117150890347
0.834061135371179
```

HyperTuning parameter (Bagging Model)

```
{'bootstrap': True, 'bootstrap_features': False, 'max_samples': 1.0, 'n_estimators': 1
0}
```

Out[220]:
```
BaggingClassifier(base_estimator=RandomForestClassifier())
```

Confusion Matrix for the training data(Bagging Model)

Out[222]:
```
array([[307,  25],
       [ 11, 724]], dtype=int64)
```

Accuracy Score of Training Data(Bagging Model)

Out[223]:
```
0.9662605435801312
```

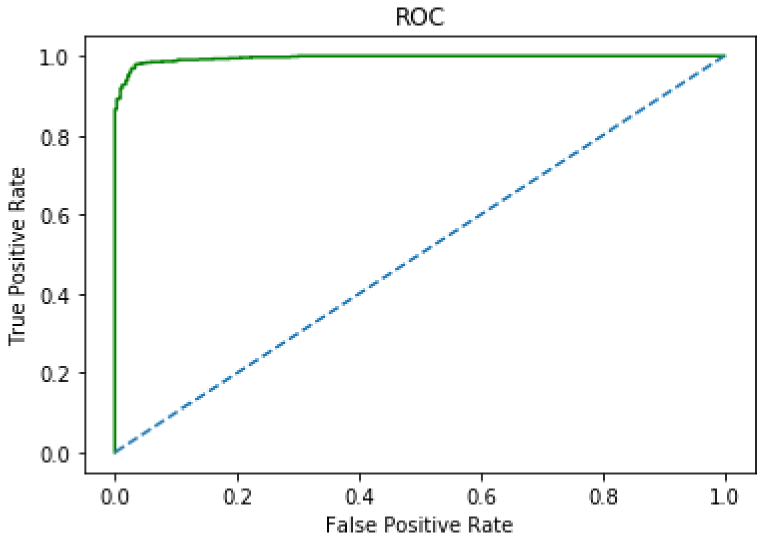
Classification Report of training data (Bagging Model)

```
              precision    recall  f1-score   support

           0       0.97      0.92      0.94       332
           1       0.97      0.99      0.98       735

    accuracy                           0.97      1067
   macro avg       0.97      0.95      0.96      1067
weighted avg       0.97      0.97      0.97      1067
```

```
Bag_train_precision  0.89
Bag_train_recall  0.88
Bag_train_f1  0.88
```

AUC and ROC of Training Data((Bagging Model)

```
Area under Curve is 0.9548664043930826
```



Performance Evaluation on Test data(Bagging Model)

Confusion Matrix Test Data(Bagging Model)

Out[227]:
```
array([[ 88,  42],
       [ 37, 291]], dtype=int64)
```

Test Data Accuracy (Bagging Model)

Out[228]:
```
0.8275109170305677
```

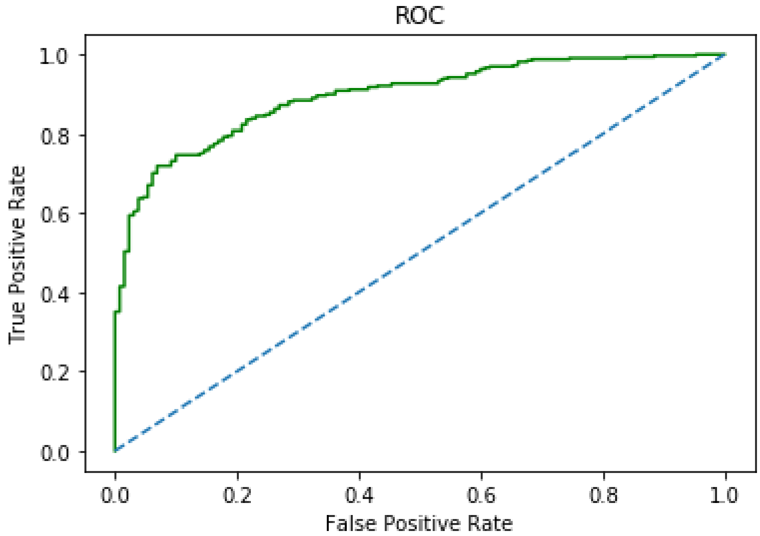
Classification Report of Test Data(Bagging Model)

```
              precision    recall  f1-score   support

           0       0.70      0.68      0.69       130
           1       0.87      0.89      0.88       328

    accuracy                           0.83       458
   macro avg       0.79      0.78      0.79       458
weighted avg       0.83      0.83      0.83       458
```

```
Bag_test_precision  0.89
Bag_test_recall  0.88
Bag_test_f1  0.87
```

AUC and ROC for the Test data((Bagging Model)

```
Area under Curve is 0.7820590994371482
```



(Bagging Model)

Out[232]:

	Bagging Train	Bagging Test
Accuracy	0.97	0.83
AUC	0.95	0.78
Recall	0.88	0.88
Precision	0.89	0.89
F1 Score	0.88	0.87

Bagging Model Final comments

- Basic Bagging Model having training and test score is 96.2 and 83.4% respectively .
- Tuning Bagging Model Model having training and test score is 96.6 and 82.7% respectively .
- Both Basic Model and Tuning Model are overfitting

Ada Boost

Out[52]: AdaBoostClassifier(base_estimator=RandomForestClassifier())

Basic Model Score (Ada Boost)

```
Training 0.9990627928772259
Testing 0.8144104803493449
```

HyperTuning parameter (AdaBoost Model)

```
{'algorithm': 'SAMME.R', 'learning_rate': 0.8, 'n_estimators': 50}
```

Out[242]: AdaBoostClassifier(base_estimator=RandomForestClassifier(), learning_rate=0.8)

Predicting the Training and Testing data(ADB)

Performance Evaluation on Training data(ADB)

Confusion Matrix of Training Data(ADB)

Out[245]:
```
array([[331,   1],
       [  0, 735]], dtype=int64)
```

Accuracy Score of Training Data(ADB)

Out[246]: 0.9990627928772259

Classification Report of Training Data(ADB)

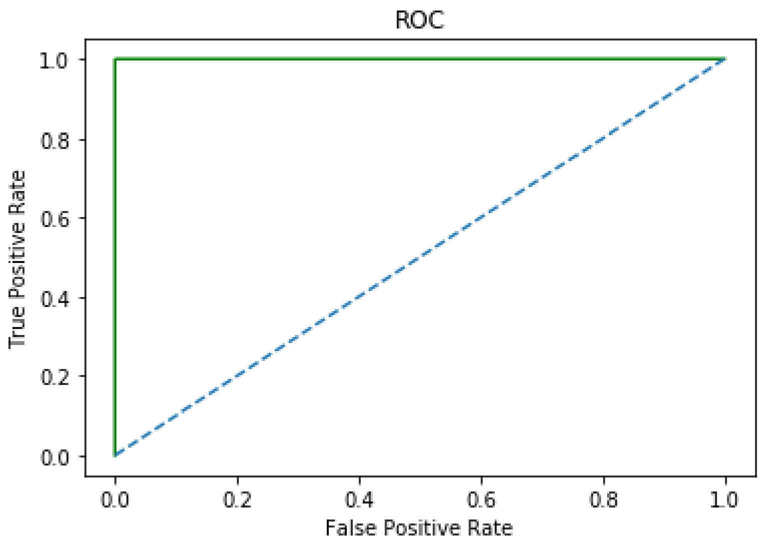
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       332
           1       1.00      1.00      1.00       735

    accuracy                           1.00      1067
   macro avg       1.00      1.00      1.00      1067
weighted avg       1.00      1.00      1.00      1067
```

Training Matrics (ADB)

```
ADB_train_precision  1.0
ADB_train_recall  1.0
ADB_train_f1  1.0
```

AUC and ROC of Training Data(ADB)

```
Area under Curve is 0.9984939759036144
```



Performance Evaluation on Test data(ADB)

Confusion Matrix Test Data(ADB)

```
Out[250]: array([[ 84,  46],
       [ 40, 288]], dtype=int64)
```

Accuracy Score of Test Data(ADB)

```
Out[251]: 0.8122270742358079
```

Classification Report of Test Data(ADB)

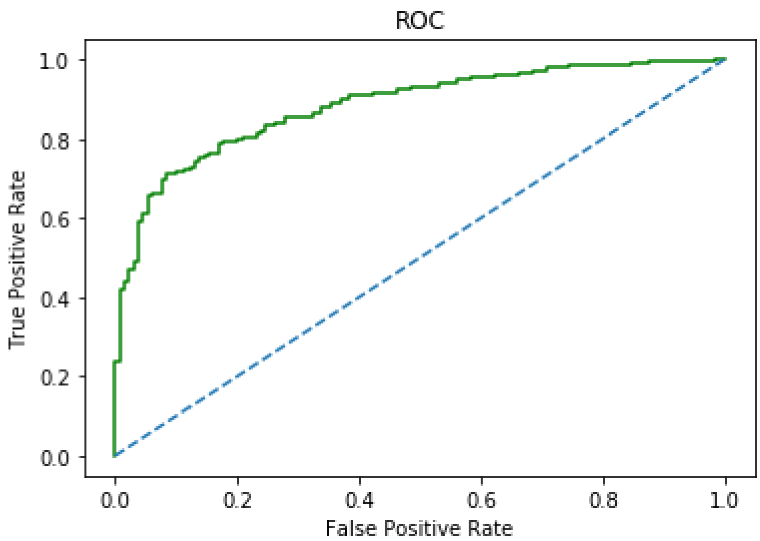
```
              precision    recall  f1-score   support

           0       0.68      0.65      0.66       130
           1       0.86      0.88      0.87       328

    accuracy                           0.81       458
   macro avg       0.77      0.76      0.77       458
weighted avg       0.81      0.81      0.81       458
```

Test Data Matrics(ADB)

```
ADB_test_precision  0.88
ADB_test_recall  0.87
ADB_test_f1  0.86
```

AUC and ROC of Test Data(ADB)

```
Area under Curve is 0.7621013133208254
```



Train and Test Performance (ADB)

Out[258]:

	ADBTrain	ADB Test
Accuracy	1.0	0.81
AUC	1.0	0.76
Recall	1.0	0.87
Precision	1.0	0.88
F1 Score	1.0	0.86

Ada Boost Model Final comments

- Basic Ada Boost Model having training and test score is 99.9 and 81.2% respectively .
- Both Basic and Tuning Ada Boost Model having training and test score is 99.9 and 81.2% respectively .
- Both Basic and Tuning Model are overfitting

XGBoost

Out[49]:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

Basic Model Score (XGBoost)

```
Training 0.9915651358950328
Testing 0.8013100436681223
```

HyperTuning parameter (XG Boost)

```
{'gamma': 10, 'learning_rate': 0.6, 'max_depth': 4, 'n_estimators': 25}
```

```
Out[271]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                        colsample_bynode=1, colsample_bytree=1, gamma=10, gpu_id=-1,
                        importance_type='gain', interaction_constraints='',
                        learning_rate=0.6, max_delta_step=0, max_depth=4,
                        min_child_weight=1, missing=nan, monotone_constraints='()',
                        n_estimators=25, n_jobs=0, num_parallel_tree=1, random_state=0,
                        reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
                        tree_method='exact', validate_parameters=1, verbosity=None)
```

Predicting the Training and Testing data(XGB)

Confusion Matrix of Training Data(XGB)

```
Out[273]: array([[233,  99],
                 [ 66, 669]], dtype=int64)
```

Accuracy Score of Training Data(XGB)

```
Out[274]: 0.845360824742268
```

Classification Report of Training Data(XGB)

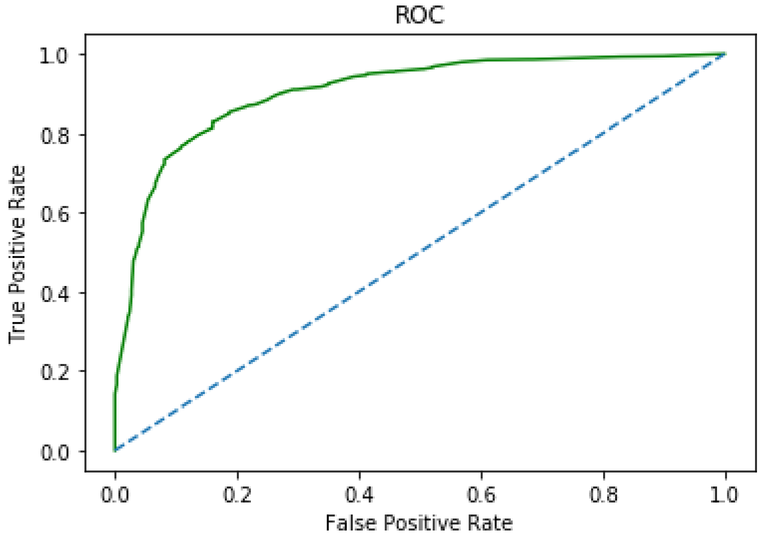
```
              precision    recall  f1-score   support

           0       0.78      0.70      0.74       332
           1       0.87      0.91      0.89       735

    accuracy                           0.85      1067
   macro avg       0.83      0.81      0.81      1067
weighted avg       0.84      0.85      0.84      1067
```

Training Matrics (XGB)

```
XGB_train_precision  0.91
XGB_train_recall  0.89
XGB_train_f1  0.87
```

AUC and ROC of Training Data(XGB)

```
Area under Curve is 0.8060056552741579
```



Performance Evaluation on Test data(XGB)

Confusion Matrix Test Data(XGB)

Out[278]:
```
array([[ 91,  39],
       [ 43, 285]], dtype=int64)
```

Accuracy Score of Test Data(XGB)

Out[279]:
```
0.8209606986899564
```

Classification Report of Test Data(XGB)

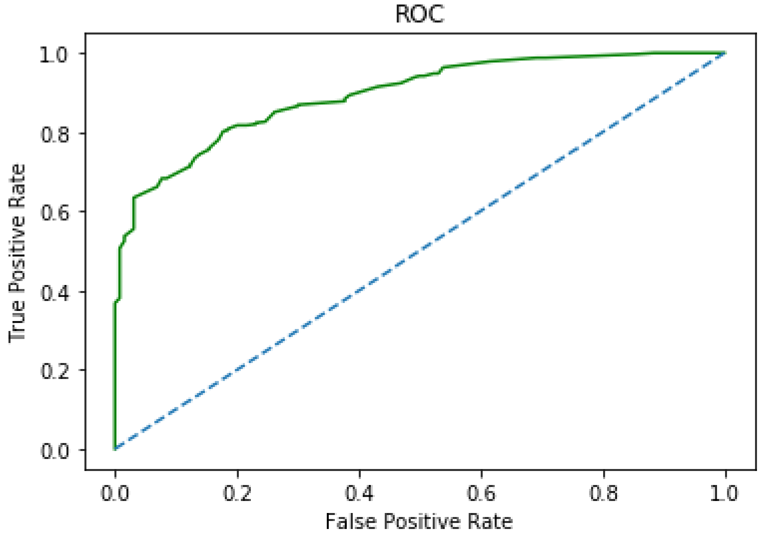
```
              precision    recall  f1-score   support

           0       0.68      0.70      0.69       130
           1       0.88      0.87      0.87       328

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

Test Data Matrics(XGB)

```
XGB_test_precision  0.87
XGB_test_recall  0.87
XGB_test_f1  0.88
```

AUC and ROC of Test Data(XGB)

```
Area under Curve is 0.7844512195121951
```



Train and Test Performance (XGB)

Out[284]:

	XGBTrain	XGB Test
Accuracy	0.85	0.82
AUC	0.81	0.78
Recall	0.89	0.87
Precision	0.91	0.87
F1 Score	0.87	0.88

XGBoost Final comments

- Basic XGBoost Model having training and test score is 99.9 and 81.2% respectively .
- Tuning Ada Boost Model having training and test score is 84.5 and 82.0% respectively .
- Basic Model is overfitting while Tuning Model is right fit Model

1.7) Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model. Final Model: Compare the models and write inference which model is best/optimized.

Out[313]:

	Logistic Train	Logistic Test	LDA Train	LDA Test	KNN Train	KNN Test	SVM Train	SVM Test	NB Train	NB Test	RF Train	RFTest	BagTrain	Ba
Accuracy	0.84	0.82	0.84	0.82	1.0	0.82	0.84	0.82	0.83	0.83	0.84	0.83	0.97	
AUC	0.80	0.77	0.81	0.78	1.0	0.76	0.81	0.78	0.89	0.88	0.79	0.79	0.95	
Recall	0.89	0.88	0.89	0.88	1.0	0.88	0.89	0.88	0.88	0.87	0.88	0.88	0.88	
Precision	0.91	0.89	0.90	0.88	1.0	0.90	0.90	0.88	0.88	0.89	0.91	0.89	0.89	
F1 Score	0.87	0.87	0.88	0.88	1.0	0.86	0.88	0.88	0.88	0.88	0.86	0.88	0.88	

Comparison Analysis

- Tunned KNN ,Ada Boositng , Bagging are overfitting Models.
- Logistic ,LDA,SVM,Naive Bayes ,Random Forest,XGBoost are right fit Model
- In the dataset Target variables are not equally distributed ,it is the ratio of 70 and 30 , So we need to focus on f1 score
- If we see the f1 score as well As Accuracy , it is almost same for the all the right fit Models
- It we compare the AUC score Naive Byes Model is outstanding Model out of all right fit Models .
- AUC Score of Naive Bayes is 90 % while rest of right fit Models having 80% approximtely. So Naive Bayes is the well optimized model

1.8) Based on these predictions, what are the insights?

- Hague , Europe and Blair are there important variables
- People whose have Economic household and Economic National condition is 3 and 4,mostly vote for Labour party
- People whose have economic National condition is 5 rarely vote for conservative party
- People who has political knowlege 1 averse to do voting.
- whose blair score is 4 mostly favoured labour party candidate
- whose Hague score is 2 mostly favoured labour party candidate
- People who have moderate(1-6 score) view towards European Integrity favour the labour party
- people who has score in European Intergrity is 11 given slightly more favour to conservative party as compare to labour party candidate

2.1) Find the number of characters, words and sentences for the mentioned documents

```
[nltk_data] Downloading package inaugural to
[nltk_data]     C:\Users\kuldipwadhwa\AppData\Roaming\nltk_data...
[nltk_data]   Package inaugural is already up-to-date!
```

Out[118]: True

2.1)a Find the number of characters, words and sentences for the mentioned documents(1941-Roosevelt.txt)

2.1)a Find the number of characters for the mentioned document(1941-Roosevelt.txt)

Out[64]: 7571

```
Number of characters in the 1941-Roosevelt speech 7571
```

2.1)a Find the number of words for the mentioned document(1941-Roosevelt.txt)

Out[67]:
```
0    1360
Name: word_count, dtype: int64
```

```
Number of words in the 1941-Roosevelt speech is 1360
```

2.1)a Find the number of Sentences for the mentioned document(1941-Roosevelt.txt)

- Important Note:- We are assuming that Sentences are stopped with dot .

Out[69]:
```
0    69
Name: sentence, dtype: int64
```

```
Number of Sentence in the 1941-Roosevelt speech is 69
```

2.1)b Find the number of characters, words and sentences for the mentioned documents(1961-Kennedy.txt)

2.1)b Find the number of characters for the mentioned document(1961-Kennedy.txt)

Out[71]: 7618

```
Number of characters in the 1961-Kennedy speech 7618
```

2.1)b Find the number of words for the mentioned document(1961-Kennedy.txt)

Out[74]:
```
0    1390
Name: word_count, dtype: int64
```

```
Number of words in the 1961-Kennedy speech 1s 1390
```

2.1)b Find the number of Sentences for the mentioned document(1961-Kennedy.txt)

- Important Note:- We are assuming that Sentences are stopped with dot .

```
Out[76]: 0    69
Name: sentence, dtype: int64
```

```
Number of sentences in the 1961-Kennedy speech 1s 69
```

2.1)c Find the number of characters, words and sentences for the mentioned documents(1973-Nixon.txt)

2.1)c Find the number of characters for the mentioned document(1973-Nixon.txt)

```
Out[78]: 9991
```

```
Number of characters in the 1973-Nixon speech 9991
```

2.1)c Find the number of words for the mentioned document(1973-Nixon.txt)

```
Out[81]: 0    1819
Name: word_count, dtype: int64
```

```
Number of words in the 1973-Nixon speech 1s 1819
```

2.1)c Find the number of Sentences for the mentioned document(1973-Nixon.txt)

- Important Note:- We are assuming that Sentences are stopped with dot .

```
Out[83]: 0    70
Name: sentence, dtype: int64
```

```
Number of senetences in the 1973-Nixon speech is 70
```

2.2) Remove all the stopwords from the three speeches.

2.2) Checking which are stop words so we can crosscheck our result

```
['a', 'about', 'above', 'after', 'again', 'against', 'ain', 'all', 'am', 'an', 'and',
'any', 'are', 'aren', "aren't", 'as', 'at', 'be', 'because', 'been', 'before', 'bein
g', 'below', 'between', 'both', 'but', 'by', 'can', 'couldn', "couldn't", 'd', 'did',
'didn', "didn't", 'do', 'does', 'doesn', "doesn't", 'doing', 'don', "don't", 'down',
'during', 'each', 'few', 'for', 'from', 'further', 'had', 'hadn', "hadn't", 'has', 'ha
sn', "hasn't", 'have', 'haven', "haven't", 'having', 'he', 'her', 'here', 'hers', 'her
self', 'him', 'himself', 'his', 'how', 'i', 'if', 'in', 'into', 'is', 'isn', "isn't",
'it', "it's", 'its', 'itself', 'just', 'll', 'm', 'ma', 'me', 'mightn', "mightn't", 'm
ore', 'most', 'mustn', "mustn't", 'my', 'myself', 'needn', "needn't", 'no', 'nor', 'no
t', 'now', 'o', 'of', 'off', 'on', 'once', 'only', 'or', 'other', 'our', 'ours', 'ours
elves', 'out', 'over', 'own', 're', 's', 'same', 'shan', "shan't", 'she', "she's", 'sh
ould', "should've", 'shouldn', "shouldn't", 'so', 'some', 'such', 't', 'than', 'that',
"that'll", 'the', 'their', 'theirs', 'them', 'themselves', 'then', 'there', 'these',
'they', 'this', 'those', 'through', 'to', 'too', 'under', 'until', 'up', 've', 'very',
'was', 'wasn', "wasn't", 'we', 'were', 'weren', "weren't", 'what', 'when', 'where', 'w
hich', 'while', 'who', 'whom', 'why', 'will', 'with', 'won', "won't", 'wouldn', "would
n't", 'y', 'you', "you'd", "you'll", "you're", "you've", 'your', 'yours', 'yourself',
'yourselves']
```

2.2) Remove all the stopwords from the three speeches.

2.2)a Remove all the stopwords from 1941-Roosevelt.txt

2.2)a Number of stopwords from 1941-Roosevelt.txt

Out[108]:
```
0    632
Name: stopwords, dtype: int64
```

2.2)a Removal of stopwords from 1941-Roosevelt.txt

2.2)a After Removal of stopwords from 1941-Roosevelt.txt

Out[118]:

	Text	stopwords
0	national day inauguration since people renewed...	0

Removal of punctuations(1941-Roosevelt.txt)

Converting to lower case (1941-Roosevelt.txt)

2.2)b Remove all the stopwords from 1961-Kennedy.txt

2.2)b Number of stopwords from 1961-Kennedy.txt

```
Out[177]: 0    0
Name: stopwords, dtype: int64
```

2.2)Removal of stopwords from 1961-Kennedy.txt

2.2)After Removal of stopwords from 1961-Kennedy.txt

```
Out[125]: 0    0
Name: stopwords, dtype: int64
```

2.2)Removal of punctuations from (1961-Kennedy.txt)

2.2)Converting to lower case (1961-Kennedy.txt)

2.2)c Remove all the stopwords from 1973-Nixon.txt

2.2)Number of stopwords from 1973-Nixon.txt

```
Out[179]: 0    70
Name: stopwords, dtype: int64
```

2.2)Removal of stopwords from 1973-Nixon.txt

2.2)After Removal of stopwords from 1973-Nixon.txt

```
Out[151]: 0    0
Name: stopwords, dtype: int64
```

Removal of punctuations from from 1973-Nixon.txt

Converting to lower case(1973-Nixon.txt)

2.3) Which word occurs the most number of times in his inaugural address for each president? Mention the top three words. (after removing the stopwords)

2.3)a Which word occurs the most number of times in his inaugural address for 1941-Roosevelt.txt Mention the top three words. (after removing the stopwords)

Important Note

- Most Number of words has been calcuated after removing the stop words,punctuations,converting to lower case

```
Out[183]: nation    11
          know      10
          spirit     9
          dtype: int64
```

2.3)b Which word occurs the most number of times in his inaugural address for 1961-Kennedy.txt Mention the top three words. (after removing the stopwords)

Important Note

- Most Number of words has been calcuated after removing the stop words,punctuations,converting to lower case
- let ,us are not in the stop words , I have not added these two words (let ,us) in stop words . while it may be subjective whether we should include or not . For my personal view Let us world make more sense rather than removing it

```
Out[184]: let      16
          us       12
          world     8
          dtype: int64
```

2.3)c Which word occurs the most number of times in his inaugural address for 1973-Nixon.txt Mention the top three words. (after removing the stopwords)

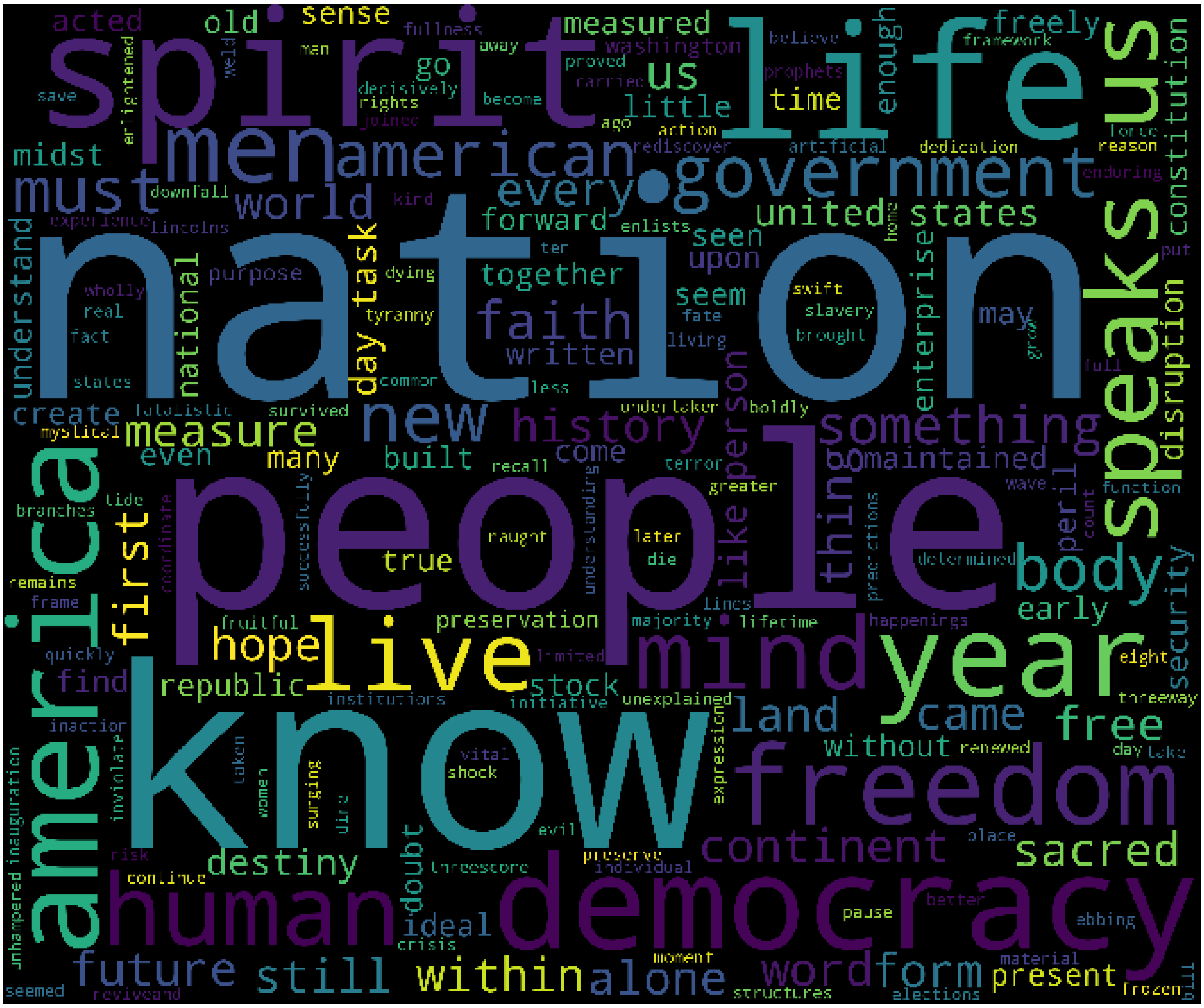
Important Note

- Most Number of words has been calcuated after removing the stop words,punctuations,converting to lower case
- let ,us are not in the stop words , I have not added these two words (let ,us) in stop words . while it may be subjective whether we should include or not . For my personal view Let us make peace make more sense rather than removing it

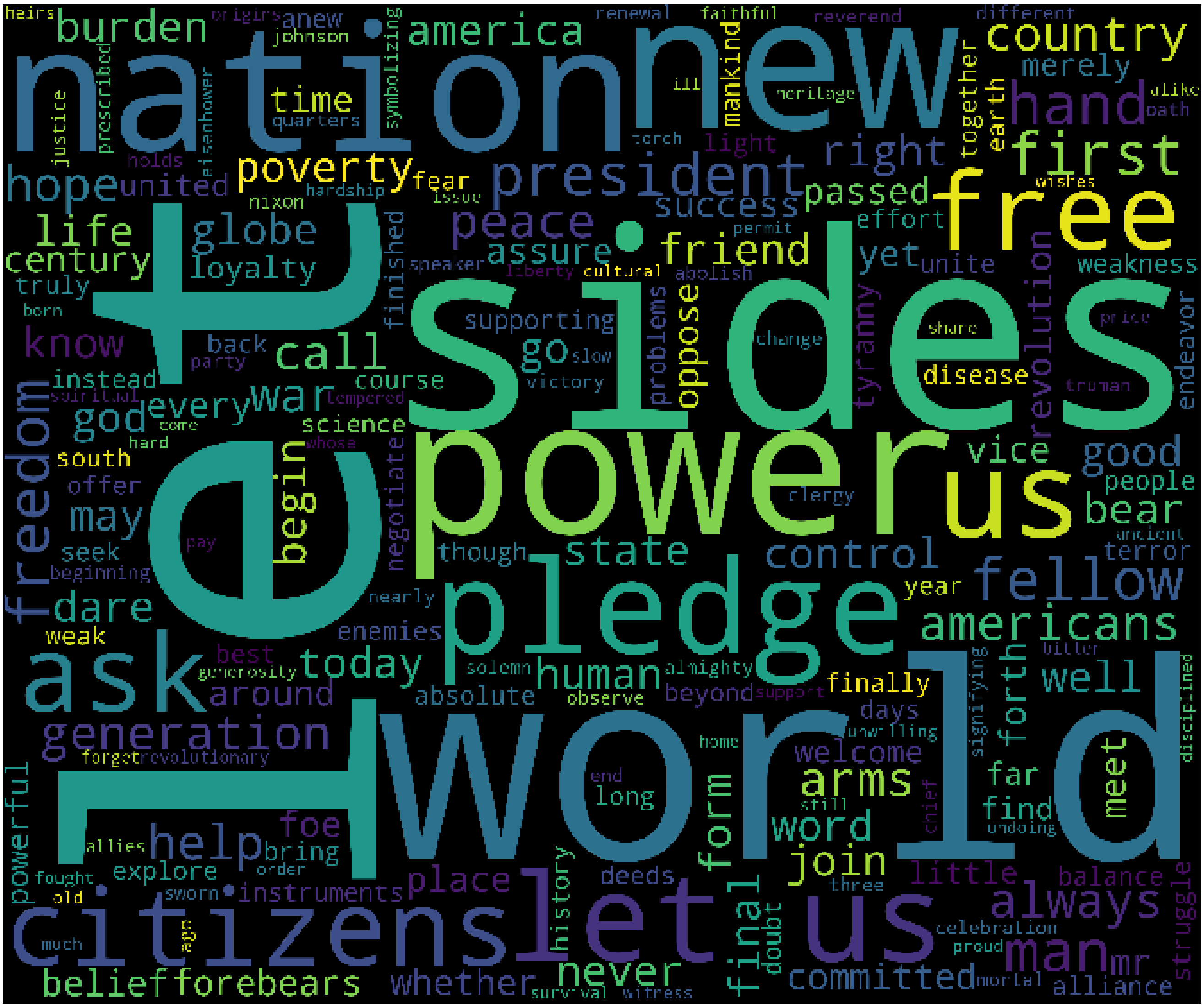
```
Out[185]: us       26
          let      22
          peace    19
          dtype: int64
```

2.4) Plot the word cloud of each of the three speeches. (after removing the stopwords)

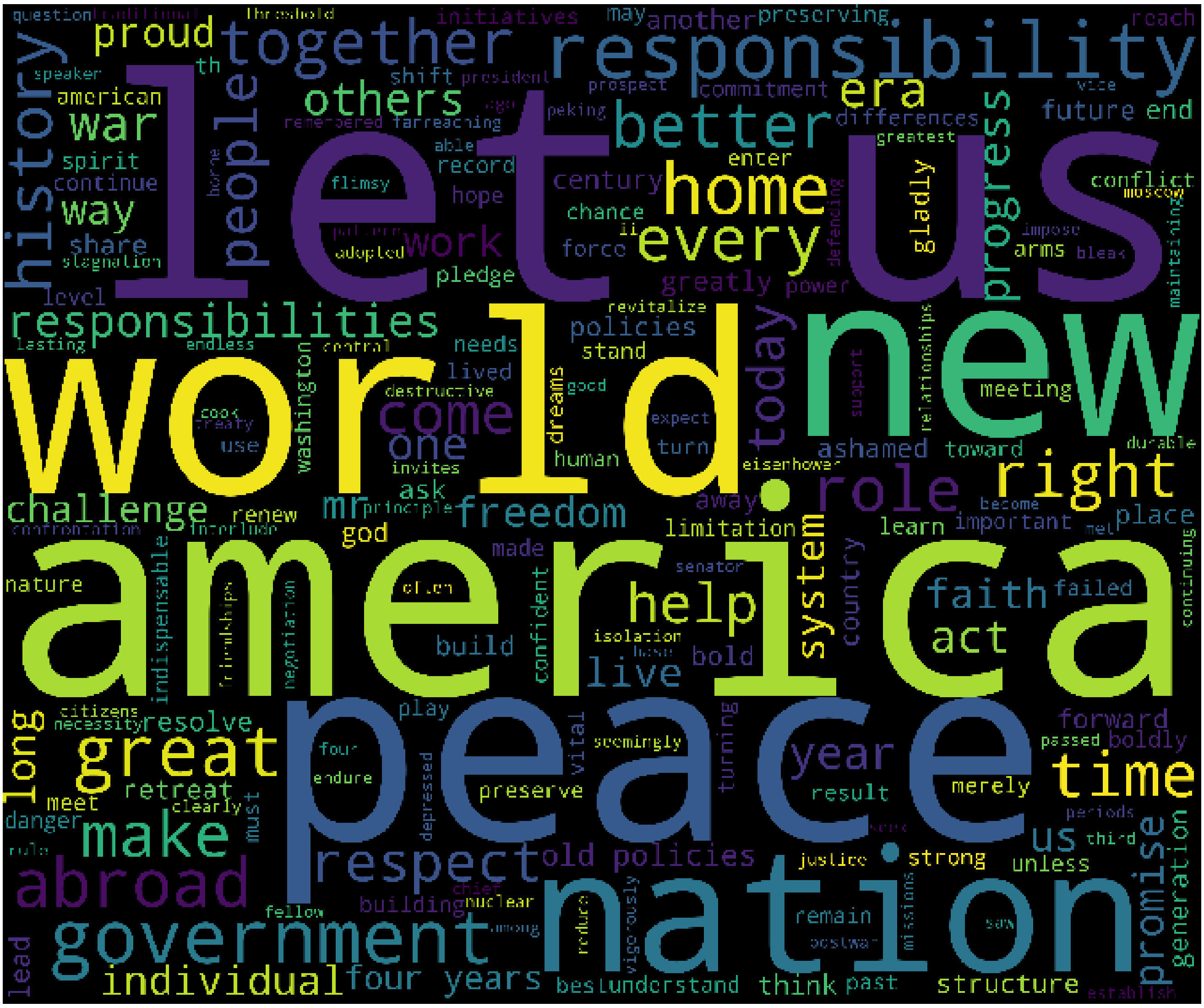
2.4)a Plot the word cloud of each of the for 1941-Roosevelt speeches.



2.4)b Plot the word cloud for 1961-Kennedy speeches.



2.4)c Plot the word cloud for 1973-Nixon.txt speeches.


let us
world
new
america
peace
nation
responsibility
together
proud
history
people
others
better
era
home
every
war
american
spirit
continue
way
share
work
progress
responsibilities
today
role
right
come
one
challenge
freedom
help
system
faith
act
great
long
make
abroad
respect
old policies
government
individual
four years
time
promise
generation
year
forward
boldly
future end
conflict
arms
gladly
century
force
greatly
power
policies
stand
needs
lived
meeting
ashamed
toward
place
important
learn
limitation
away
mr
god
ask
renew
nature
build
live
bold
country
confident
indispensable
washington
dreams
human
turn
resolve
retreat
meet
danger
preserve
result
merely
us
unless
strong
justice
remain
structure
past
think
best understand
building
lead
play
vital
seemingly
turning
depressed
must
clearly
citizens
necessity
fellow
nuclear
pledge
hope
record
chance
enter
differences
greatest
commitment
preserving
another
initiatives
may
reach
question
threshold
speaker
shift
president
peking
prospect
able
remembered
farreaching
flimsy
adopted
revitalize
level
stagnation
lasting
endless
control
destructive
good
expect
invites
principle
made
failed
senator
isolation
eisenhower
durable
become
continuing
confrontation
interlude
nature
four
endure
passed
third
periods
establish

/