Project – Machine Learning

JULY 19, 2022

PGP DSBA JAN 22A,
Great Learning
Authored by: Sharjil Shah

Contents

Problem 1:

You are hired by one of the leading news channels CNBE who wants to analyze recent elections. This survey was conducted on 1525 voters with 9 variables. You have to build a model, to predict which party a voter will vote for on the basis of the given information, to create an exit poll that will help in predicting overall win and seats covered by a particular party.

Dataset for Problem: Election_Data.xlsx

Variable Name	**Description**
vote	Party choice: Conservative or Labour
age	in years
economic.cond.national	Assessment of current national economic conditions, 1 to 5 (1 – poor, 5- great)
economic.cond.househol d	Assessment of current household economic conditions, 1 to 5 (1 – poor, 5- great)
Blair	Assessment of the Labour leader, 1 to 5
Hague	Assessment of the Conservative leader, 1 to 5
Europe	Assessment of the Conservative leader, 1 to 5
political.knowledge	Knowledge of parties' positions on European integration, 0 to 3
gender	female or male

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

Let's take a look at the head and tail of the dataset.

	Unnamed: 0	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	1	Labour	43	3	3	4	1	2	2	female
1	2	Labour	36	4	4	4	4	5	2	male
2	3	Labour	35	4	4	5	2	3	2	male
3	4	Labour	24	4	2	2	1	4	0	female
4	5	Labour	41	2	2	1	1	6	2	male

The dataset has no null values, there are 8 variables that are integers and 2 variables are objects.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 10 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   Unnamed: 0               1525 non-null   int64 
 1   vote                     1525 non-null   object
 2   age                      1525 non-null   int64 
 3   economic.cond.national   1525 non-null   int64 
 4   economic.cond.household  1525 non-null   int64 
 5   Blair                    1525 non-null   int64 
 6   Hague                    1525 non-null   int64 
 7   Europe                   1525 non-null   int64 
 8   political.knowledge      1525 non-null   int64 
 9   gender                   1525 non-null   object
dtypes: int64(8), object(2)
memory usage: 119.3+ KB
```

a) Descriptive statistics of the dataset

	Unnamed: 0	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	1525.000000
mean	763.000000	54.182295	3.245902	3.140328	3.334426	2.746885	6.728525	1.542295
std	440.373894	15.711209	0.880969	0.929951	1.174824	1.230703	3.297538	1.083315
min	1.000000	24.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000
25%	382.000000	41.000000	3.000000	3.000000	2.000000	2.000000	4.000000	0.000000
50%	763.000000	53.000000	3.000000	3.000000	4.000000	2.000000	6.000000	2.000000
75%	1144.000000	67.000000	4.000000	4.000000	4.000000	4.000000	10.000000	2.000000
max	1525.000000	93.000000	5.000000	5.000000	5.000000	5.000000	11.000000	3.000000

b) Checking for null values:

```
Unnamed: 0                 False
vote                       False
age                        False
economic.cond.national     False
economic.cond.household    False
Blair                      False
Hague                      False
Europe                     False
political.knowledge        False
gender                     False
dtype: bool
```

```
Unnamed: 0                 0
vote                       0
age                        0
economic.cond.national     0
economic.cond.household    0
Blair                      0
Hague                      0
Europe                     0
political.knowledge        0
gender                     0
dtype: int64
```

c) Skewness of the dataset

```
age                         0.139800
economic.cond.national     -0.238474
economic.cond.household    -0.144148
Blair                      -0.539514
Hague                       0.146191
Europe                     -0.141891
political.knowledge        -0.422928
dtype: float64
```

d) Inference:

- With the problem statement we know that the target variable is 'Vote' from the dataset.
- The head and tail of the dataset tells us that there are 2 main parties for whom the voters are voting for and they are namely: 'Labour' and 'Conservative'.
- The dataset has 10 unique columns, out of which 2 are objects and 8 are integers. Column "Unnamed:0" is an index column and will be dropped while performing EDA.
- From the descriptive statistics we can see that the youngest voter is of the age 24, 50% of the voters are of the age 53 and the oldest voter is 93 years old
- Labour party seems to be bagging more number of votes and the most number of voters are females
- Variables 'economic.cond.national', 'economic.cond.household', 'Blair', 'Hague', 'Europe' and 'political.knowledge' are ordinal variables.
- 50% of the voters have assessed 'Blair' who is the leader of Labour Party to be at 4 which is higher than that of 'Hague' who is the leader of Conservative Party

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

a) Exploratory Data Analysis:

The dataset has 10 columns and 1525 rows.

```
There are total 1525 rows and 10 columns in the dataset
```

Column "Unnamed: 0" should be dropped since it does not have any significance in this study Snippet below shows the head and tail after dropping column "Unnamed: 0"

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	Labour	43	3	3	4	1	2	2	female
1	Labour	36	4	4	4	4	5	2	male
2	Labour	35	4	4	5	2	3	2	male
3	Labour	24	4	2	2	1	4	0	female
4	Labour	41	2	2	1	1	6	2	male

Although we know from section 1.1 that there are no null values, the snippet below proves the point further. (Note: Please check the code for this inference from jupyter notebook)

```
Unnamed: 0                 0
vote                       0
age                        0
economic.cond.national     0
economic.cond.household    0
Blair                      0
Hague                      0
Europe                     0
political.knowledge        0
gender                     0
dtype: int64
```

Similarly, let's also check the data types of each variable. Variable "vote" and "gender" are objects and the rest are integers.

```
Unnamed: 0                  int64
vote                       object
age                         int64
economic.cond.national      int64
economic.cond.household     int64
Blair                       int64
Hague                       int64
Europe                      int64
political.knowledge         int64
gender                     object
dtype: object
```

The dataset has 8 duplicate records which will be dropped as they do not add any value to the study

```
Number of duplicate rows = 8
```

Let's take a look at the shape of the dataset after dropping the duplicated records and column "Unnamed: 0"

```
Before (1525, 9)
After (1517, 9)
```

As seen in section 1.1, variables "economic.cond.national", "economic.cond.household", "Blair", "Hague", "Europe" and "political.knowledge" are ordinal variables and must be converted to object data type. Info of the variables after converting the variables.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1517 entries, 0 to 1524
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   vote                     1517 non-null   object
 1   age                      1517 non-null   int64
 2   economic.cond.national   1517 non-null   int64
 3   economic.cond.household  1517 non-null   int64
 4   Blair                    1517 non-null   int64
 5   Hague                    1517 non-null   int64
 6   Europe                   1517 non-null   int64
 7   political.knowledge      1517 non-null   int64
 8   gender                   1517 non-null   object
dtypes: int64(7), object(2)
memory usage: 150.8+ KB
```

Let's check the unique values in the categorical/ object variables

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


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

Since variable "vote" is our target variable and it has 2 categories.

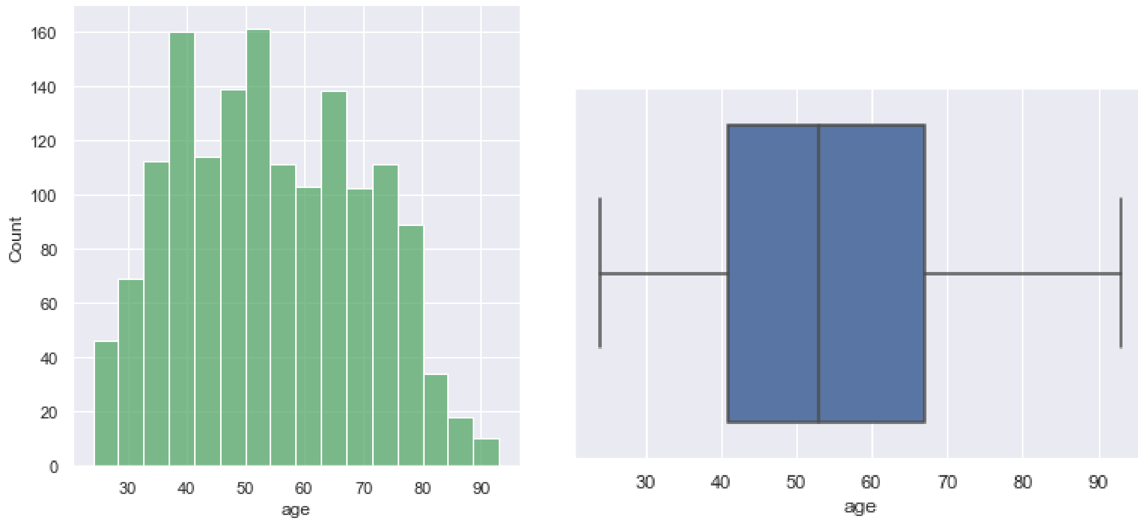
i) <u>Inferences:</u>

- By doing initial EDA, we can say that the original data set had 1525 rows and 10 columns
- There are 2 variables whose datatype is "object" and these are categorical variables
- The other variables "economic.cond.national", "economic.cond.household", "Blair", "Hague", "Europe" and "political.knowledge" are also ordinal/ categorical and hence their data type has been converted

- The dataset does not have any null or missing values and none of the categorical variables have a as "?" or "blank"
- There were 8 duplicate records which were dropped as they do not add any value to our analysis
- The percentage of votes are not balanced between the two parties where 69.68% of the voters voted for Labour party and only 30.32% of the voters voted for Conservative party.

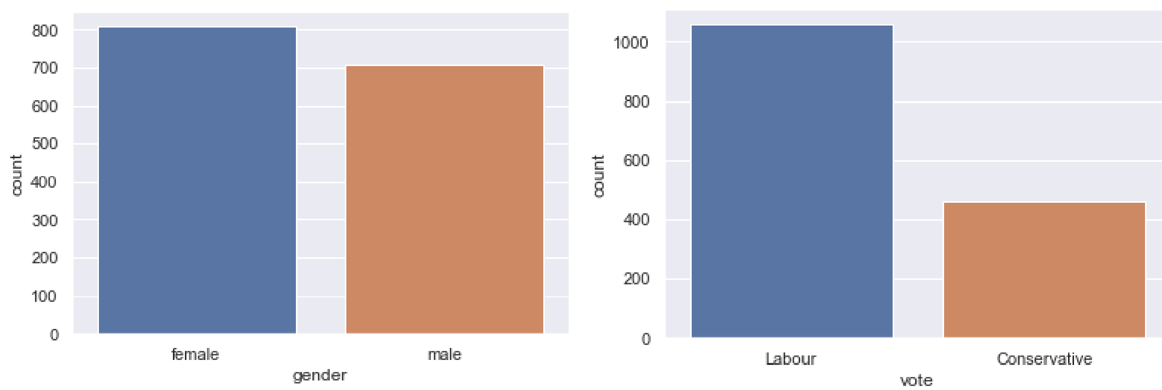
b) Univariate Analysis:

i) Figure below shows the Univariate Analysis using Distplot and Boxplot of variable "age"



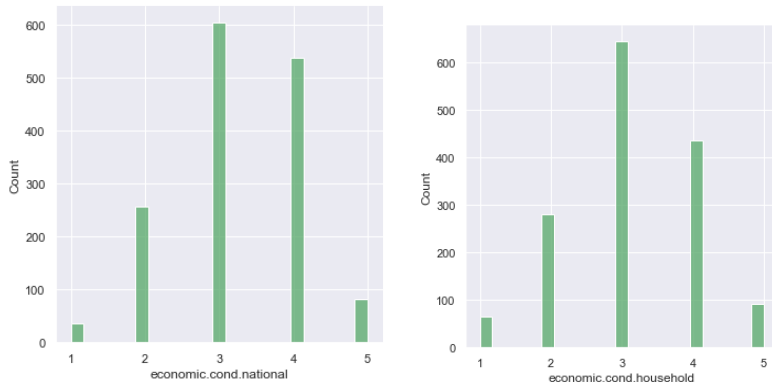
There is a normal distribution in variable "age". Most of the voters are found to be between the age of 40 to 80.

ii) Figure below shows the Univariate Analysis using Countplot of variables "vote" and "gender"



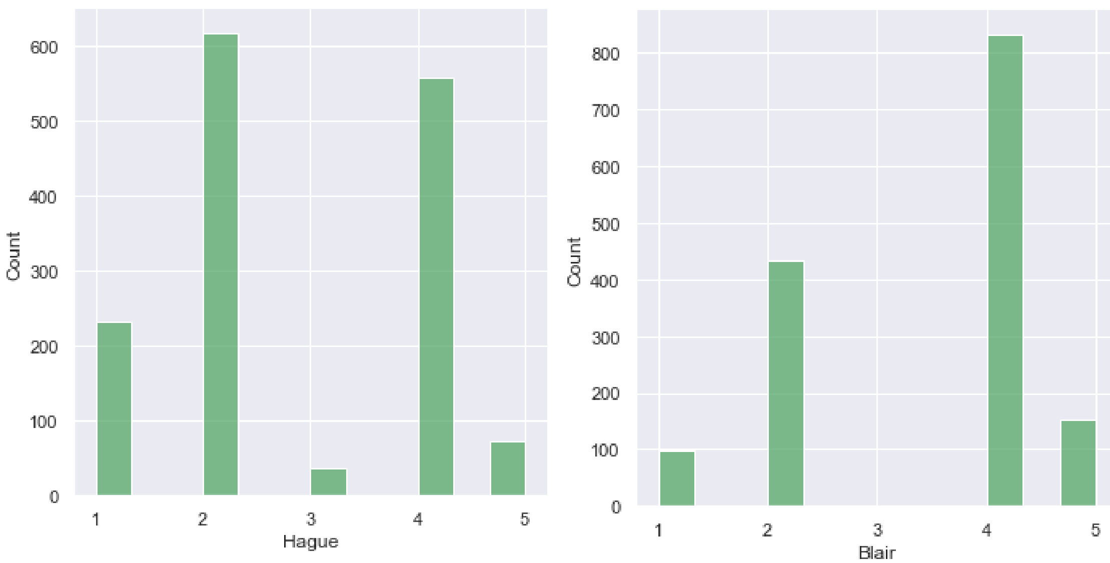
We can infer that the Labour party is being favored more by the voters. The ratio of female to male is almost the same with female voters being more than male

iii) Figure below shows the Univariate Analysis using Countplot of variables "economic.cond.national" and "economic.cond.household"



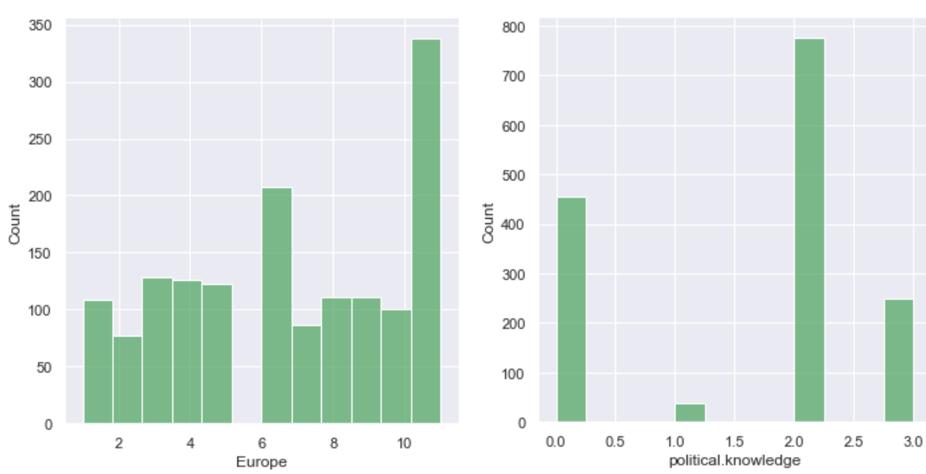
Variable “economic.cond.national” and “economic.cond.household” show that most of the voters have rated these two scales as 3 and 4 which is a moderate rating.

iv) Figure below shows the Univariate Analysis using Countplot of variables “Blair” and “Hague”



Most of the voters have voted “Blair” as 4 compared to “Hague”
Many voters have rated “Hague” as 2 compared to “Blair”

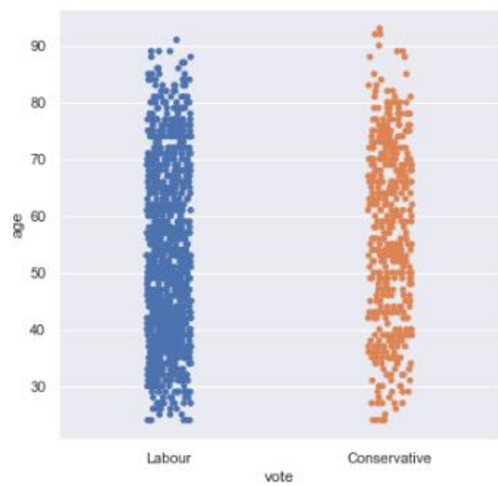
v) Figure below shows the Univariate Analysis using Countplot of variables “Europe” and “political.knowledge”



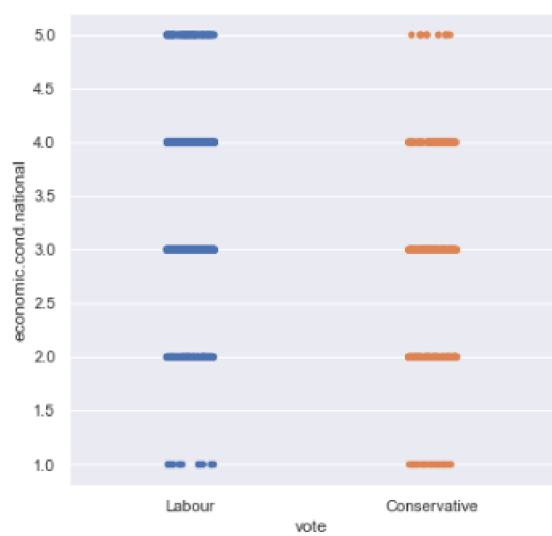
In a scale of 1 to 11, most of the voters have voted that the European integration is between 2 to 10 with maximum as 11. Hence, most of them have an inclination that the parties represent 'Eurosceptic' sentiment.
In a scale of 0 to 3, most of the voters have voted the Political knowledge to be 2 which is moderate.

c) Bivariate Analysis

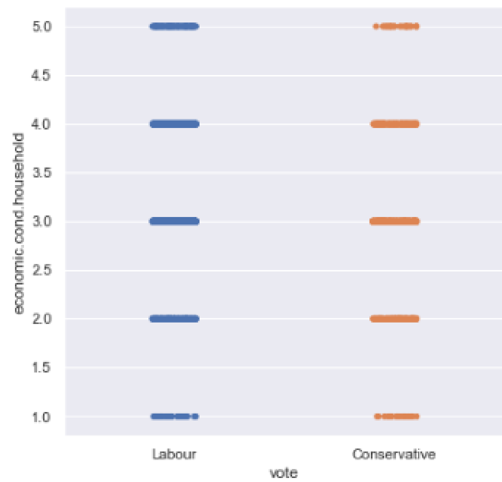
vi) Figure below shows the Bivariate Analysis using Strip plot which are taken from jupyter notebook



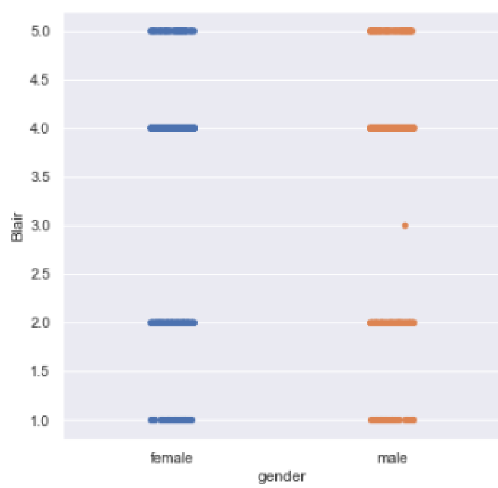
Nearly similar trend. Since there are more voters for Labour party, the strip looks denser for Labour party. One key difference would be that most of the voters above the age of 90 have voted for Conservative party.



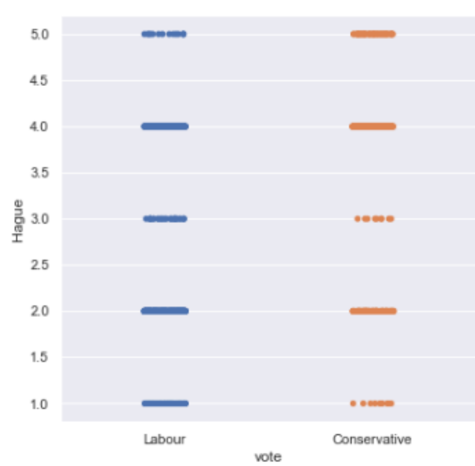
We can see that most of the voters have rated Labour party as 5 compared to Conservative Party for their assessment on current national economic conditions. Otherwise the trend is nearly the same.



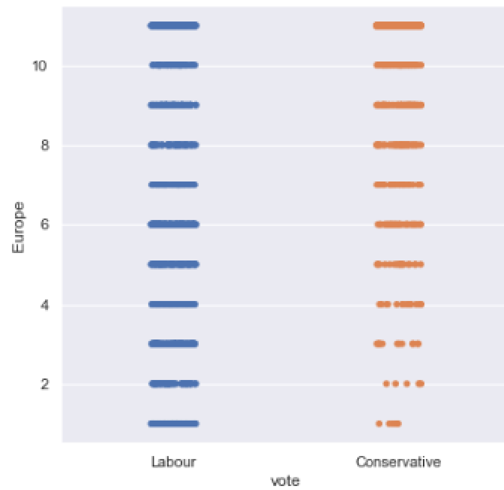
Trend is nearly the same. Except that more voters have rated Labour party to be a 5 when it comes to an assessment on economic household conditions.



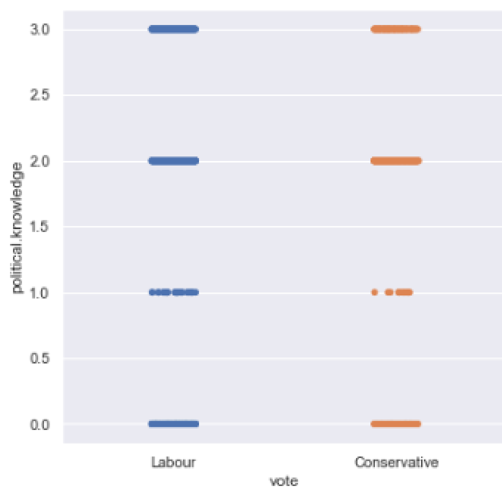
Trend is the same with one exception that none of the female voters rated Blair as 3 but few male voters have rated him as 3.



Trend is nearly the same.



Many voters have rated Labour party between the scale of 2 to 5 compared to the Conservative Party. Otherwise the trend is the same.



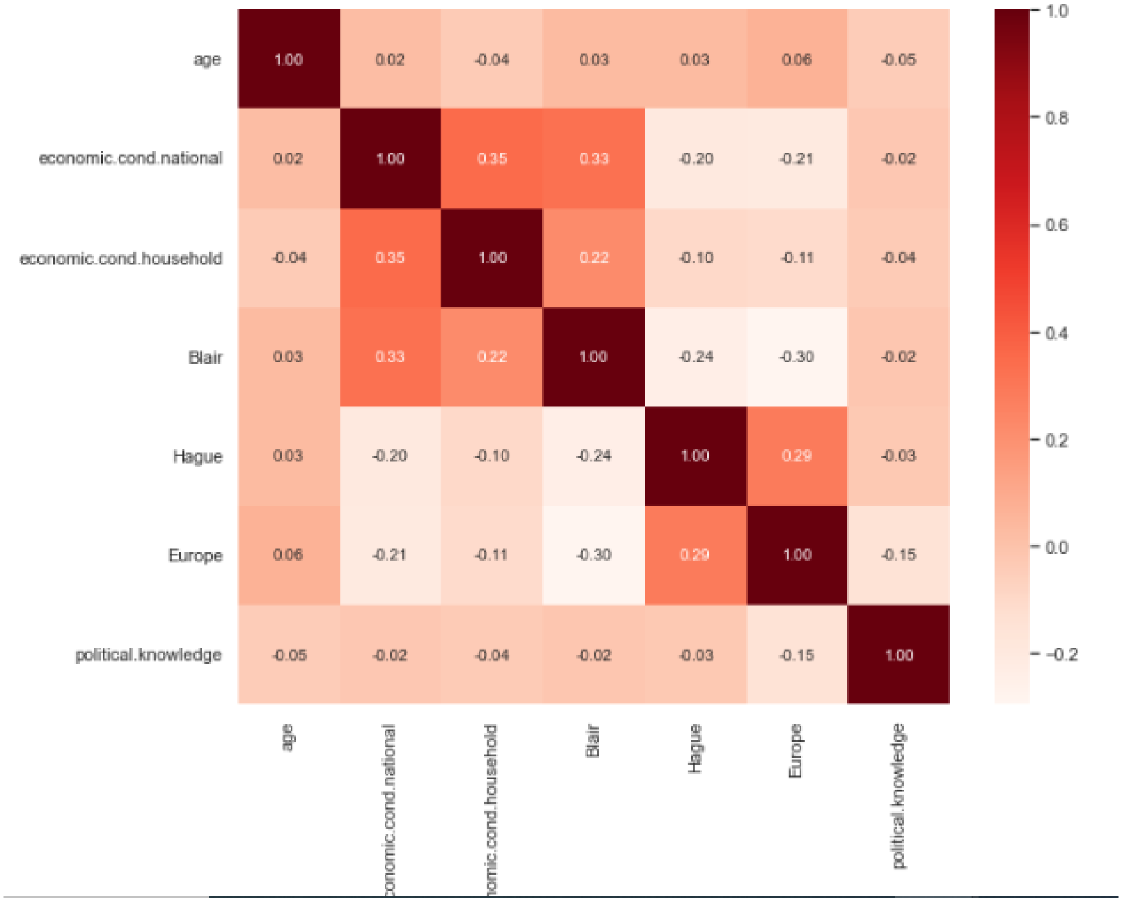
On a scale of 0 to 3, Labour party seems to have more voters rating them as 1 compared to Conservative Party. Otherwise the trend is the same.

vii) Figure below shows the Correlation Matrix and Heat Map which are taken from the jupyter notebook

Correlation Matrix before converting all the variables except “age” to categorical variables

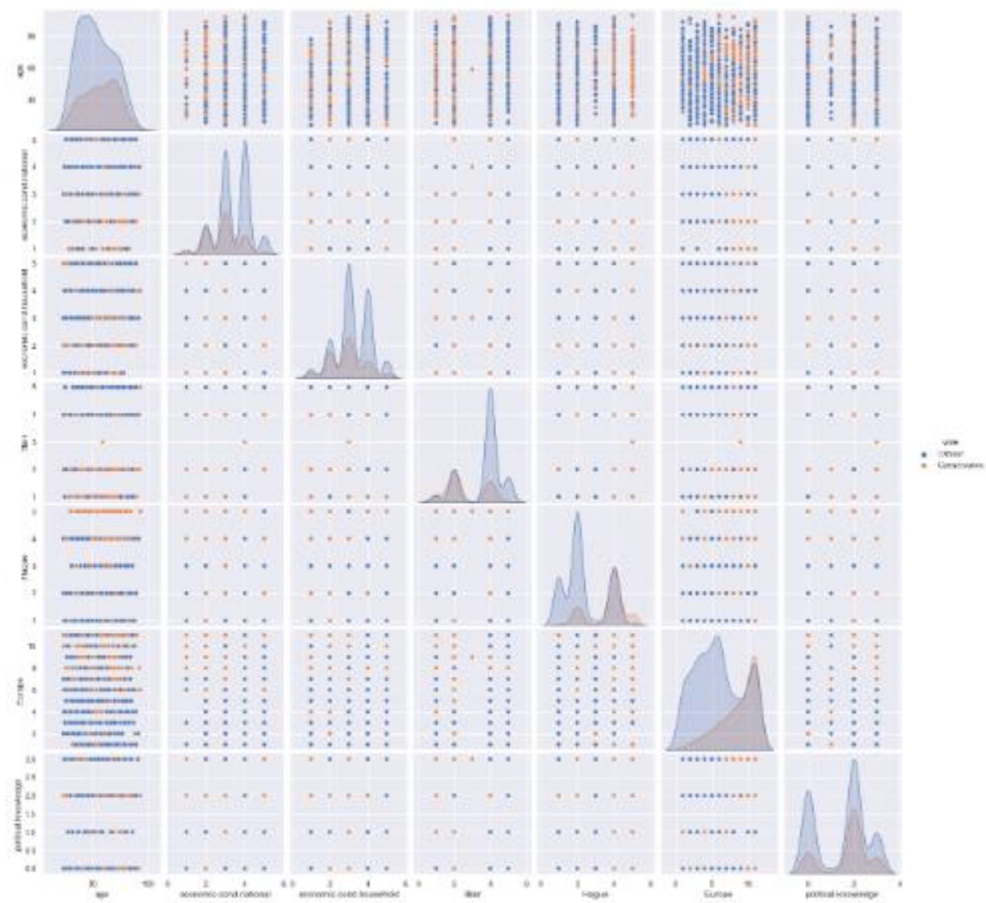
	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge
age	1.000000	0.018687	-0.038868	0.032084	0.031144	0.064562	-0.046598
economic.cond.national	0.018687	1.000000	0.347687	0.326141	-0.200790	-0.209150	-0.023510
economic.cond.household	-0.038868	0.347687	1.000000	0.215822	-0.100392	-0.112897	-0.038528
Blair	0.032084	0.326141	0.215822	1.000000	-0.243508	-0.295944	-0.021299
Hague	0.031144	-0.200790	-0.100392	-0.243508	1.000000	0.285738	-0.029906
Europe	0.064562	-0.209150	-0.112897	-0.295944	0.285738	1.000000	-0.151197
political.knowledge	-0.046598	-0.023510	-0.038528	-0.021299	-0.029906	-0.151197	1.000000

Heat Map before converting all the variables except “age” to categorical variables



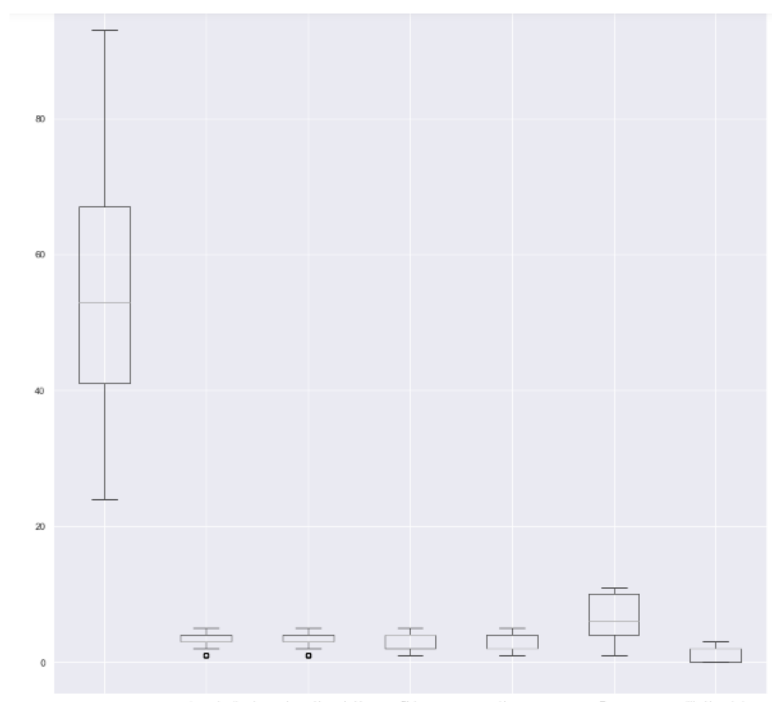
- The heat map shows that there is no high correlation between any of the variables as most of the values are under 0.35
- There is negative correlation between age and & political knowledge and "economic.cond.household"
- Variables "economic.cond.national" and "economic.cond.household" have the highest correlation of 0.35

viii) Figure below shows the Pair Plot which is taken from the jupyter notebook.



No correlation is found between the variables

ix) Figure below shows if there are outliers present which is taken from the jupyter notebook



- No outlier found in variable “age”
- Some outliers are found in variables “economic.cond.household” and “economic.cond.household” which can be checked in Univariate analysis done in jupyter notebook. Since those are ordinal variables, we will not be treating them.

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).

a) Data Encoding:

- Variable “vote” & “gender” contain string values. To perform the analysis for this dataset, we would be converting the string values to the integer data type as modelling cannot take string / object data types.
- For converting variable to numerical data we will use pd.Categorical().codes function
- Variable “gender” has ‘male’ and ‘female’ which will be converted to 1 and 0 respectively.
- Variable “vote” has “Labour” and “Conservative” which will be converted to 1 and 0 respectively

Snippet from jupypter notebook shows that that column “gender” has been changed to numerical data

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	vote_Labour	gender_male
0	43	3	3	4	1	2	2	1	0
1	36	4	4	4	4	5	2	1	1
2	35	4	4	5	2	3	2	1	1
3	24	4	2	2	1	4	0	1	0
4	41	2	2	1	1	6	2	1	1

Copy all the predictor variables into X dataframe
Copy target into the y dataframe.

b) Scaling:

- Scaling is done on continuous variables in a dataset with different unit of measures.
- All variables are either categorical or ordinal except for variable “age”.
- For Logistic regression, LDA and Naïve Baye’s model we need not perform any scaling, however, for KNN it is necessary to scale the data, as it a distance-based algorithm (typically based on Euclidean distance).
- For KNN, the variables will be scaled using the min max scaler.

c) Train and Test Split:

The data will be split using train_test_split() with random state = 1 and test size = 0.30.

```
1    754
0    307
Name: vote_Labour, dtype: int64
```

```
1    303
0    153
Name: vote_Labour, dtype: int64
```

1.4 Apply Logistic Regression and LDA (Linear Discriminant Analysis). Interpret the inferences of both models.

a) Apply Logistic Regression:

The data has been pre-processed and has been split to train and test data with test size = 0.30 in the previous sections.

Step 1: We will apply Logistic Regression with below parameters.

Step 2: Check the model score by using model.score()

```
Accuracy score  for Logistic regression  train variable 0.8369462770970783
 Accuracy score  for Logistic regression  test variable 0.8289473684210527
```

Step 3: Check confusion matrix

Train data:

```
array([[198, 109],
       [ 64, 690]], dtype=int64)
```

Test data:

```
array([[110,  43],
       [ 35, 268]], dtype=int64)
```

Heat Map of Confusion Matrix:

Step 4: Check classification matrix

```
Classification report of Logistic Regression Training Data

              precision    recall  f1-score   support

           0       0.76      0.64      0.70       307
           1       0.86      0.92      0.89       754

    accuracy                           0.84      1061
   macro avg       0.81      0.78      0.79      1061
weighted avg       0.83      0.84      0.83      1061

Classification report of Logistic Regression Testing Data

              precision    recall  f1-score   support

           0       0.76      0.72      0.74       153
           1       0.86      0.88      0.87       303

    accuracy                           0.83       456
   macro avg       0.81      0.80      0.81       456
weighted avg       0.83      0.83      0.83       456
```

- Hyper parameters were added to see if the model behave differently in training and testing data. By using solver = liblinear, there was some differences found in the models.
- The accuracy of model in training set is 0.84 and on testing set is 0.83, which is good and very close to each other.
- The recall of Conservative party is better on Testing data whereas the recall of Labour party is better on Training data
- Overall it is a good model and there is no over fitting found.

b) Apply Linear Discriminant Analysis:

Step 1: We will apply LDA to the training and testing data.

```
▾ LinearDiscriminantAnalysis
LinearDiscriminantAnalysis()
```

Step 2: Check the model score by using model.score()

```
Accuracy score  for LDA  train variable 0.8341187558906692

Accuracy score  for LDA  test variable 0.8333333333333334
```

Step 3: Check confusion matrix

Train data:

```
array([[200, 107],
       [ 69, 685]], dtype=int64)
```

Test data:

```
array([[111,  42],
       [ 34, 269]], dtype=int64)
```

Heat Map of confusion matrix

Step 4: Check classification matrix

Classification report of LDA Training Data

	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

Classification report of LDA Testing Data

	precision	recall	f1-score	support
0	0.77	0.73	0.74	153
1	0.86	0.89	0.88	303
accuracy			0.83	456
macro avg	0.82	0.81	0.81	456
weighted avg	0.83	0.83	0.83	456

- The accuracy of model in training set and testing set is the same which is 0.83.
- The recall of Conservative party is better on Testing data whereas the recall of Labour party is better on Training data
- Overall the model is performing well.

Comparison of two models:

While comparing both these models, we find both results are almost same, but LDA works better since the recall with LDA is slightly better on Testing data.

1.5 Apply KNN Model and Naïve Bayes Model. Interpret the inferences of each model.

a) Apply KNN model:

Step 1: Scale the data

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
0	-0.716161	-0.278185	-0.148020	0.565802	-1.419969	-1.437338	0.423832	-0.936736
1	-1.162118	0.856242	0.926367	0.565802	1.014951	-0.527684	0.423832	1.067536
2	-1.225827	0.856242	0.926367	1.417312	-0.608329	-1.134120	0.423832	1.067536
3	-1.926617	0.856242	-1.222408	-1.137217	-1.419969	-0.830902	-1.421084	-0.936736
4	-0.843577	-1.412613	-1.222408	-1.988727	-1.419969	-0.224465	0.423832	1.067536

Step 2: We will apply KNN with below parameters.

```
▾ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=9)
```

Steps 3: Check the model score by using model.score().

```
Accuracy score  for KNN train variable 0.8416588124410933
Accuracy score  for KNN test variable 0.8223684210526315
```

Step 4: Check confusion matrix:

Train data:

```
array([[214,  93],
       [ 75, 679]], dtype=int64)
```

Test data:

```
array([[107,  46],
       [ 35, 268]], dtype=int64)
```

Heat Map of confusion matrix

Step 5: Check classification matrix

```
KNN Classfication report

Classification Report of the training data:
              precision    recall  f1-score   support

           0       0.74      0.70      0.72       307
           1       0.88      0.90      0.89       754

    accuracy                           0.84      1061
   macro avg       0.81      0.80      0.80      1061
weighted avg       0.84      0.84      0.84      1061

Classification Report of the test data:
              precision    recall  f1-score   support

           0       0.75      0.70      0.73       153
           1       0.85      0.88      0.87       303

    accuracy                           0.82       456
   macro avg       0.80      0.79      0.80       456
weighted avg       0.82      0.82      0.82       456
```

- The accuracy of model is 0.84 and 0.82 on training and testing data respectively
- The recall of Conservative party is better on Testing data and Labour is slightly better on Training data

b) Apply Gaussian Naïve Bayes model:

Step 1: We will apply Gaussian Naïve Bayes with below parameters

```
▾ GaussianNB
GaussianNB()
```

Step 2: Check the model score by using model.score()

```
Accuracy score  for NB train variable 0.8350612629594723

Accuracy score  for NB test variable 0.8223684210526315
```

Step 3: Check confusion matrix

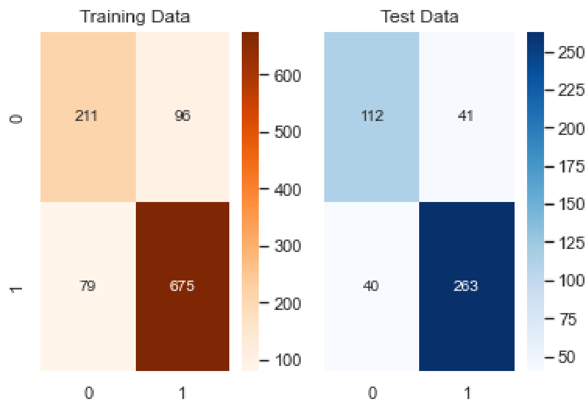
Train data:

```
array([[211,  96],
       [ 79, 675]], dtype=int64)
```

Test data:

```
array([[112,  41],
       [ 40, 263]], dtype=int64)
```

Heat Map of confusion matrix



Step 4: Check classification matrix

```
Naive Bayse Classfication report

Classification Report of the training data:
              precision    recall  f1-score   support

           0       0.73      0.69      0.71       307
           1       0.88      0.90      0.89       754

    accuracy                           0.84      1061
   macro avg       0.80      0.79      0.80      1061
weighted avg       0.83      0.84      0.83      1061


Classification Report of the test data:
              precision    recall  f1-score   support

           0       0.74      0.73      0.73       153
           1       0.87      0.87      0.87       303

    accuracy                           0.82       456
   macro avg       0.80      0.80      0.80       456
weighted avg       0.82      0.82      0.82       456
```

- The accuracy of model in training set is 0.84 and on testing set is 0.82. Hence, the model is performs better on Training data.
- The recall of Conservative party is better on Testing data whereas the recall of Labour party is better on Training data

c) Comparison of both KNN and Naive Bayes model:

- Both models are good and does not overfit or underfit. The accuracy of both models on Training data is the same, however, accuracy of KNN is better on testing data
- KNN model is better compared to Naïve Bayes after applying hyperparameters as the model has better recall.

1.6 **Model Tuning, Bagging (Random Forest should be applied for Bagging), and Boosting.**

a) Tune the model using GridSerachCV and apply to Logistic Regression:

Step 1: Tune Logistics Regression model using GridSearchCV

```
▸ GridSearchCV
▸ estimator: RandomForestClassifier
  ▸ RandomForestClassifier
```

Step 2: Best parameters and estimators

```
{'max_depth': 10, 'max_features': 1}
```

Step 3: Random forest classifier

```
▾ RandomForestClassifier
RandomForestClassifier(n_estimators=50, random_state=1)
```

Step 4: bagging with random forest classfier

```
▸ BaggingClassifier
▸ base_estimator: RandomForestClassifier
  ▸ RandomForestClassifier
```

i) Bag score for train:

```
0.9651272384542884
```

ii) Bag score for test:

```
0.831140350877193
```

iii) Boosting = Ada Boosting

```
▾ AdaBoostClassifier
AdaBoostClassifier(n_estimators=10, random_state=1)
```

iv) Train Score:

```
0.8426013195098964
```

v) Test Score:

```
0.8201754385964912
```

vi) Boosting = Gradient Boosting

```
▾ GradientBoostingClassifier
GradientBoostingClassifier(n_estimators=50, random_state=1)
```

vii) Train score:

```
0.8803016022620169
```

viii) Test score:

```
0.8289473684210527
```

ix) Bagging random forest

```
Accuracy score  for Bagging train variables  0.9651272384542884
Accuracy score  for Bagging test variables  0.831140350877193
```

x) Confusion matrix for Bagging Random Forest

```
confusion matrix Train variables for Bagging Random Forest
```

xi) Confusion matrices:

Train:

```
array([[276,  31],
       [  6, 748]], dtype=int64)
```

Test:

```
array([[103,  50],
       [ 27, 276]], dtype=int64)
```

xii) Bagging (Random Forest) Classfication report

```
Classification Report of the training data:
              precision    recall  f1-score   support

           0       0.98      0.90      0.94       307
           1       0.96      0.99      0.98       754

    accuracy                           0.97      1061
   macro avg       0.97      0.95      0.96      1061
weighted avg       0.97      0.97      0.96      1061


Classification Report of the test data:
              precision    recall  f1-score   support

           0       0.79      0.67      0.73       153
           1       0.85      0.91      0.88       303

    accuracy                           0.83       456
   macro avg       0.82      0.79      0.80       456
weighted avg       0.83      0.83      0.83       456
```

b) Boosting = Ada Boosting

i) Model score:

```
Accuracy score  for ADA Boosting train variables  0.8426013195098964
Accuracy score  for ADA Boosting test variables  0.8201754385964912
```

ii) Confusion matrix Train variables for ADA Boosting:

Train data:

```
array([[206, 101],
       [ 66, 688]], dtype=int64)
```

Test data:

```
array([[110,  43],
       [ 39, 264]], dtype=int64)
```

iii) ADA Boosting Classfication report:

```
Classification Report of the training data:
              precision    recall  f1-score   support

           0       0.76      0.67      0.71       307
           1       0.87      0.91      0.89       754

    accuracy                           0.84      1061
   macro avg       0.81      0.79      0.80      1061
weighted avg       0.84      0.84      0.84      1061


Classification Report of the test data:
              precision    recall  f1-score   support

           0       0.74      0.72      0.73       153
           1       0.86      0.87      0.87       303

    accuracy                           0.82       456
   macro avg       0.80      0.80      0.80       456
weighted avg       0.82      0.82      0.82       456
```

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.

a) Logistic Regression

i) Model Accuracy:

```
Accuracy score  for Logistic regression  train variables   0.8289473684210527
```

```
Accuracy score  for Logistic regression  train variables   0.8369462770970783
```

ii) Heat Map of confusion matrix:

confusion matrix Train and Test variables for logistic regression

iv) Classification matrix:

Train data:

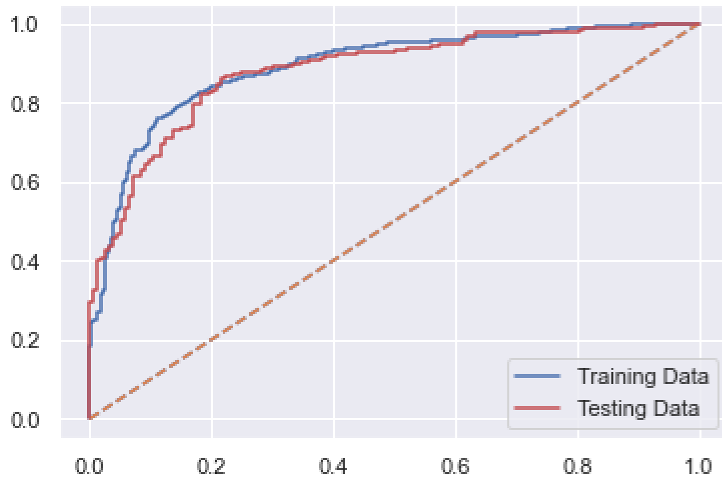
	precision	recall	f1-score	support
0	0.743494	0.651466	0.694444	307.000000
1	0.864899	0.908488	0.886158	754.000000
accuracy	0.834119	0.834119	0.834119	0.834119
macro avg	0.804197	0.779977	0.790301	1061.000000
weighted avg	0.829771	0.834119	0.830686	1061.000000

Test data:

	precision	recall	f1-score	support
0	0.765517	0.725490	0.744966	153.000000
1	0.864952	0.887789	0.876221	303.000000
accuracy	0.833333	0.833333	0.833333	0.833333
macro avg	0.815235	0.806639	0.810594	456.000000
weighted avg	0.831589	0.833333	0.832182	456.000000

v) AUC and ROC

```
AUC for the Training Data: 0.890
AUC for the Testing Data: 0.890
```



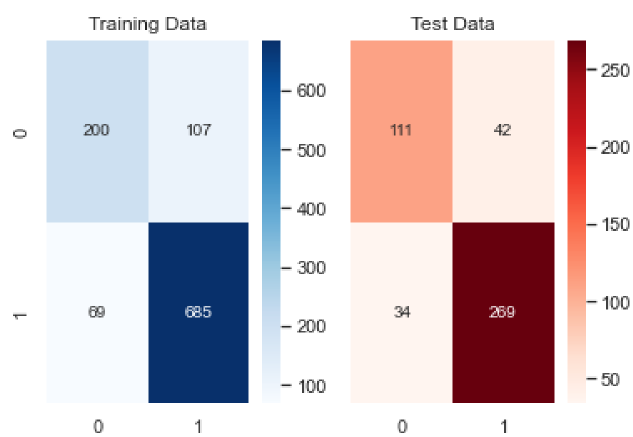
b) Linear Discrimination Analysis:

i) Model Accuracy:

```
Accuracy score  for LDA  train variables  0.8341187558906692

Accuracy score  for LDA  test variables  0.8333333333333334
```

ii) Heat Map of confusion matrix



iii) Classification matrix:

Train:

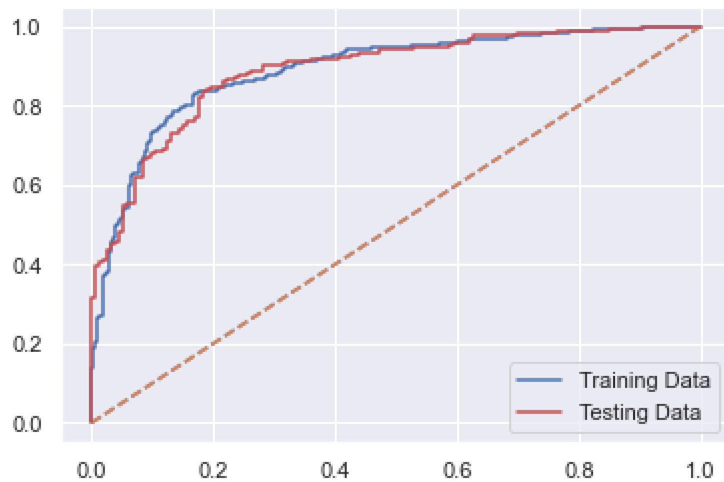
	precision	recall	f1-score	support
0	0.743494	0.651466	0.694444	307.000000
1	0.864899	0.908488	0.886158	754.000000
accuracy	0.834119	0.834119	0.834119	0.834119
macro avg	0.804197	0.779977	0.790301	1061.000000
weighted avg	0.829771	0.834119	0.830686	1061.000000

Test:

	precision	recall	f1-score	support
0	0.765517	0.725490	0.744966	153.000000
1	0.864952	0.887789	0.876221	303.000000
accuracy	0.833333	0.833333	0.833333	0.833333
macro avg	0.815235	0.806639	0.810594	456.000000
weighted avg	0.831589	0.833333	0.832182	456.000000

vi) AUC and ROC

```
AUC for the Training Data: 0.889
AUC for the Test Data: 0.888
```



c) KNN:

i) Model Accuracy

```
Accuracy score  for KNN train variable 0.8416588124410933
Accuracy score  for KNN test variable 0.8223684210526315
```

ii) Heat Map of confusion matrix

iii) Classification matrix

Train

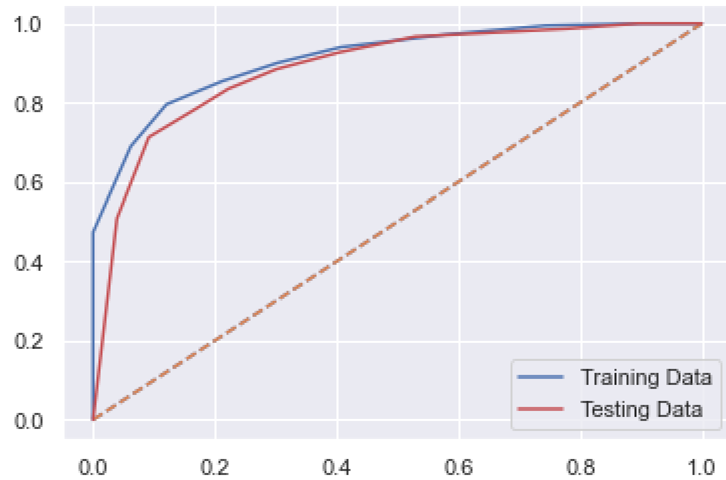
	precision	recall	f1-score	support
0	0.740484	0.697068	0.718121	307.000000
1	0.879534	0.900531	0.889908	754.000000
accuracy	0.841659	0.841659	0.841659	0.841659
macro avg	0.810009	0.798799	0.804015	1061.000000
weighted avg	0.839300	0.841659	0.840202	1061.000000

Test:

	precision	recall	f1-score	support
0	0.753521	0.699346	0.725424	153.000000
1	0.853503	0.884488	0.868720	303.000000
accuracy	0.822368	0.822368	0.822368	0.822368
macro avg	0.803512	0.791917	0.797072	456.000000
weighted avg	0.819957	0.822368	0.820640	456.000000

iv) AUC and ROC:

```
AUC for the Training Data: 0.913
AUC for the Test Data: 0.887
```

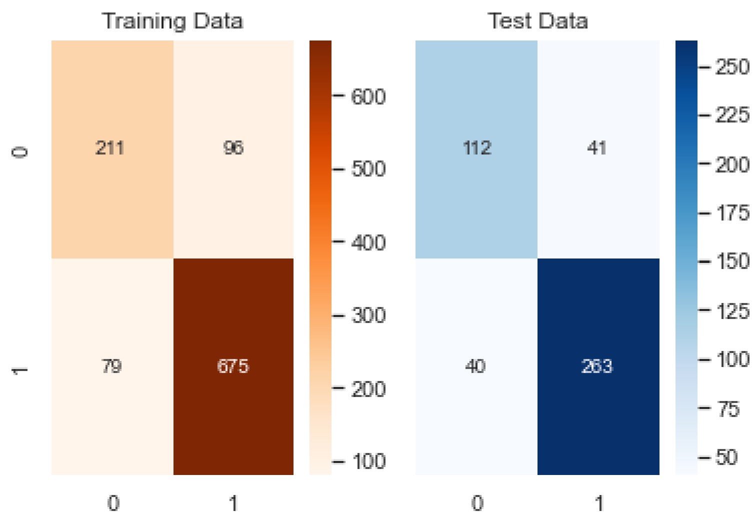


d) Naive Bayes

i) Model Accuracy

```
Accuracy score  for NB train variable 0.8350612629594723
Accuracy score  for NB test variable 0.8223684210526315
```

ii) Heat Map of confusion matrix



iii) Classification matrix

Training data:

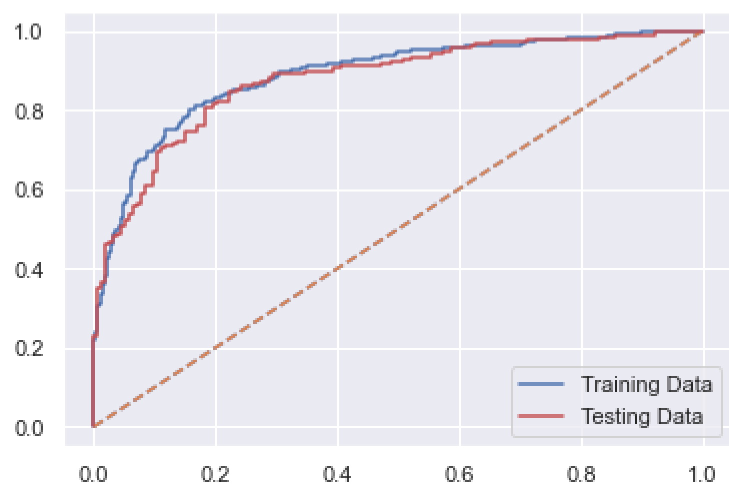
	precision	recall	f1-score	support
0	0.727586	0.687296	0.706868	307.000000
1	0.875486	0.895225	0.885246	754.000000
accuracy	0.835061	0.835061	0.835061	0.835061
macro avg	0.801536	0.791261	0.796057	1061.000000
weighted avg	0.832692	0.835061	0.833632	1061.000000

Testing data:

	precision	recall	f1-score	support
0	0.736842	0.732026	0.734426	153.000000
1	0.865132	0.867987	0.866557	303.000000
accuracy	0.822368	0.822368	0.822368	0.822368
macro avg	0.800987	0.800006	0.800492	456.000000
weighted avg	0.822087	0.822368	0.822224	456.000000

v) AUC and ROC

```
AUC for the Training Data: 0.888
AUC for the Test Data: 0.876
```



e) Ada Boost

i) Model accuracy

```
Accuracy score  for ADA Boosting train variables  0.8426013195098964
Accuracy score  for ADA Boosting test variables  0.8201754385964912
```

ii) Heat Map of confusion matrix:

iii) Classification matrix:

Training data:

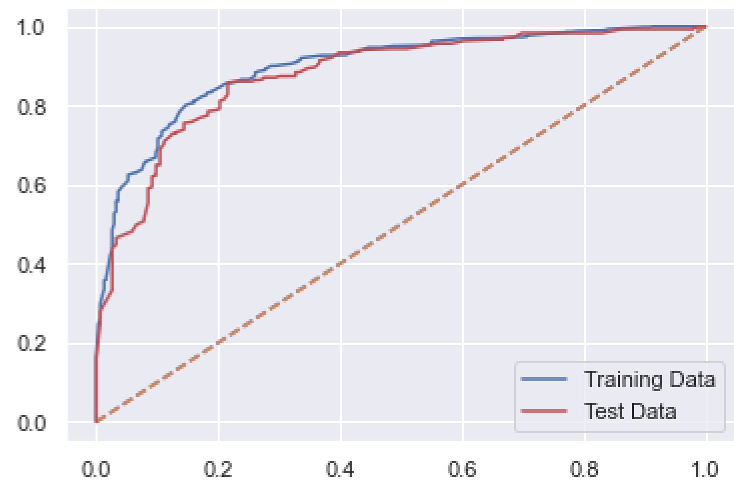
	precision	recall	f1-score	support
0	0.757353	0.671010	0.711572	307.000000
1	0.871990	0.912467	0.891769	754.000000
accuracy	0.842601	0.842601	0.842601	0.842601
macro avg	0.814671	0.791738	0.801670	1061.000000
weighted avg	0.838820	0.842601	0.839629	1061.000000

Testing data:

	precision	recall	f1-score	support
0	0.738255	0.718954	0.728477	153.000000
1	0.859935	0.871287	0.865574	303.000000
accuracy	0.820175	0.820175	0.820175	0.820175
macro avg	0.799095	0.795121	0.797025	456.000000
weighted avg	0.819108	0.820175	0.819574	456.000000

iv) AUC and ROC:

```
AUC for the Training Data: 0.898
AUC for the Test Data: 0.878
```

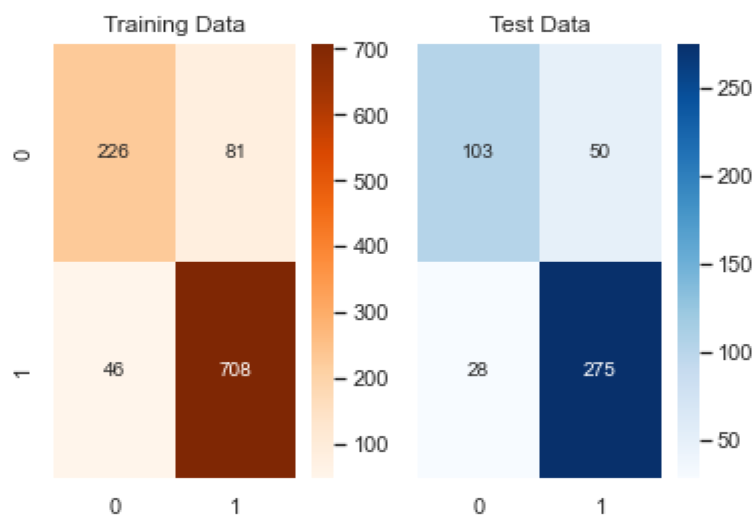


f) Gradient Boost

a) Model accuracy

```
Accuracy score  for Gradient Boosting train variables  0.8803016022620169
Accuracy score  for Gradient Boosting test variables  0.8289473684210527
```

b) Heat Map of confusion matrix



c) Classification matrix

Training data:

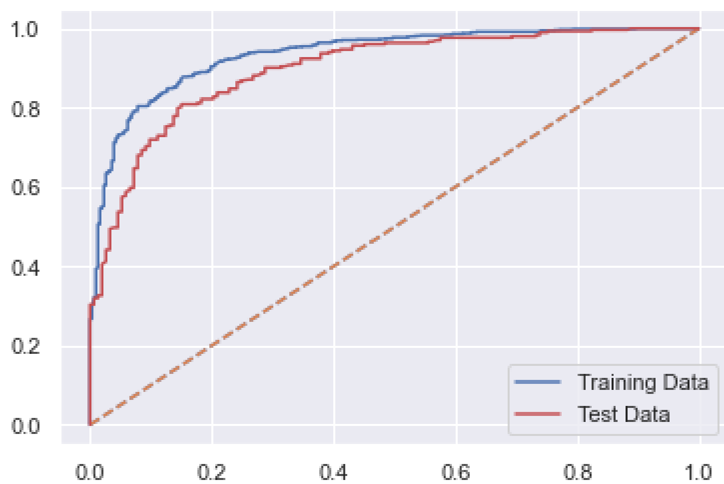
	precision	recall	f1-score	support
0	0.830882	0.736156	0.780656	307.000000
1	0.897338	0.938992	0.917693	754.000000
accuracy	0.880302	0.880302	0.880302	0.880302
macro avg	0.864110	0.837574	0.849175	1061.000000
weighted avg	0.878109	0.880302	0.878041	1061.000000

Testing data:

	precision	recall	f1-score	support
0	0.786260	0.673203	0.725352	153.000000
1	0.846154	0.907591	0.875796	303.000000
accuracy	0.828947	0.828947	0.828947	0.828947
macro avg	0.816207	0.790397	0.800574	456.000000
weighted avg	0.826058	0.828947	0.825318	456.000000

d) AUC and ROC

```
AUC for the Training Data: 0.935
AUC for the Test Data: 0.897
```



g) Bagging using Random Forest

1) Model Accuracy

```
Accuracy score  for Bagging train variables  0.9651272384542884
Accuracy score  for Bagging test variables  0.831140350877193
```

2) Heat Map of confusion matrix

3) Classification matrix

Training data:

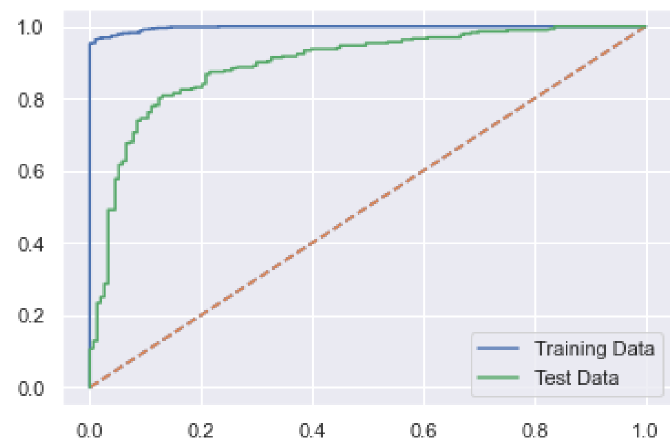
	precision	recall	f1-score	support
0	0.978723	0.899023	0.937182	307.000000
1	0.960205	0.992042	0.975864	754.000000
accuracy	0.965127	0.965127	0.965127	0.965127
macro avg	0.969464	0.945533	0.956523	1061.000000
weighted avg	0.965564	0.965127	0.964672	1061.000000

Testing data:

	precision	recall	f1-score	support
0	0.792308	0.673203	0.727915	153.00000
1	0.846626	0.910891	0.877583	303.00000
accuracy	0.831140	0.831140	0.831140	0.83114
macro avg	0.819467	0.792047	0.802749	456.00000
weighted avg	0.828401	0.831140	0.827366	456.00000

4) AUC and ROC

```
AUC for the Training Data: 0.997
AUC for the Test Data: 0.896
```



h) Comparison of Performance Metrics

	Logistic reg Train	Logistic reg Test	LDA Train	LDA Test	KNN Train	KNN Test	Naive Bayes Train	Naive Bayes Test	Bagging Train	Bagging Test	Ada Boosting Train	Ada Boosting Test	Gradient Boosting Train	Gradient Boosting Test
Accuracy	0.84	0.83	0.83	0.83	0.84	0.82	0.84	0.82	0.97	0.83	0.84	0.82	0.88	0.83
AUC	0.89	0.88	0.89	0.89	0.91	0.89	0.89	0.88	1.00	0.90	0.90	0.88	0.94	0.90
Recall	0.92	0.88	0.91	0.89	0.90	0.88	0.90	0.87	0.99	0.91	0.91	0.87	0.94	0.91
Precision	0.86	0.86	0.86	0.86	0.88	0.85	0.88	0.87	0.96	0.85	0.87	0.86	0.90	0.85
F1 Score	0.89	0.87	0.89	0.88	0.89	0.87	0.89	0.87	0.98	0.88	0.89	0.87	0.92	0.88

By comparing the performance metrics, we can conclude the following:

- Logistic Regression, LDA, KNN and Gausion Naïve Bayes are good models because they work well on both Training and Testing data with model accuracy similar across both training and testing data.
- However, LDA has better accuracy and recall and fi-score.
- Gradient Boosting and Bagging using Random Forest is not a good model because it is overfitting on training data and doesn't perform well on testing data

1.8 Based on these predictions, what are the insights.

Comparing all the Models we see that Logistic Regression, LDA, KNN and Gradient Boosting are good models, however, LDA Model gives better results.

- We observe Labour has higher possibility of winning
- Labour has higher voting possibility among all age groups except for very old people.
- Irrespective of the political knowledge levels or gender, Labour has an edge on higher votes
- Where the Eurosceptic sentiment is more, Conservative has scope for winning

Problem 2

In this particular project, we are going to work on the inaugural corpora from the nltk in Python. We will be looking at the following speeches of the Presidents of the United States of America:

1. President Franklin D. Roosevelt in 1941
2. President John F. Kennedy in 1961
3. President Richard Nixon in 1973

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

Number of words:

1. Number of words in President Franklin D. Roosevelt speech 1360
2. Number of words in President John F.Kennedy speech 1390
3. Number of words in President Richard Nixon speech 1819

Number of characters:

1. Number of characters in President Franklin D. Roosevelt speech 7571
2. Number of characters in President John F.Kennedy speech 7618
3. Number of characters in President Richard Nixon speech 9991

Number of sentences:

1. Number of sentences in President Franklin D. Roosevelt speech 67
2. Number of sentences in President John F.Kennedy speech 52
3. Number of sentences in President Richard Nixon speech 68

2.2 Remove all the stopwords from all three speeches.

1. 184 stop words were identified
2. Stop words count in Roosevelt's Speech is 730
3. Stop words count in Kennedy's Speech is 711
4. Stop words count in Nixon's Speech is 1017
5. Word count in Roosevelt's Speech after removing stop words is 604
6. Word count in Kennedy's Speech after removing stop words is 652

Word count in Nixon's Speech after removing stop words is 784

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)

In Roosevelt’s speech, below words occur more frequently

1. Nation – 11 times
2. Spirit – 9 times
3. Democracy – 9 times

In Kennedy’s speech, below words occur more frequently

World – 8 times

1. new – 7 times
2. pledge – 7 times

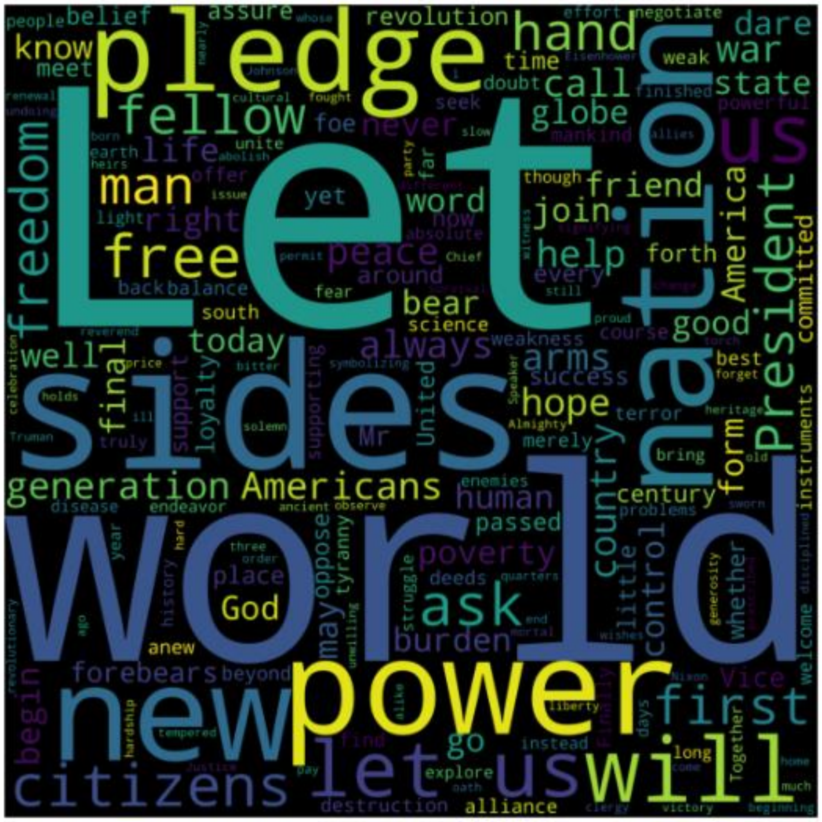
In Nixon’x speech, below words occur more frequently

1. Peace – 19 times
2. World – 16 times
3. New – 15 times
4. America – 13 times

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

A) Roosevelt:

B) Kennedy:



C) Nixon: