Machine Learning Assignment : Problem 1: Election Data

MAY 19

Machine Learning Assignment
:Suhas Pawar

Problem 1 : Election Data

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.

Data Dictionary:

Sr No	Feature	Meaning
1	Vote	Party choice: Conservative or Labour
2	age	in years
3	economic.cond.national	Assessment of current national economic conditions, 1 to 5.
4	economic.cond.household:	Assessment of current household economic conditions, 1 to 5.
5	Blair	Assessment of the Labour leader, 1 to 5.
6	Hague	Assessment of the Conservative leader, 1 to 5.
7	Europe	an 11-point scale that measures respondents' attitudes toward European integration. High scores represent 'Eurosceptic' sentiment.
8	Political knowledge	Knowledge of parties' positions on European integration, 0 to 3.
9	gender	female or male.

Question List:

- **Data Ingestion:** 11 marks

  1.1 Read the dataset. Do the descriptive statistics and do the null value condition check. Write an inference on it. (4 Marks)

  1.2 Perform Univariate and Bivariate Analysis. Do exploratory data analysis. Check for Outliers. (7 Marks)

- **Data Preparation**: 4 marks

  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). (4 Marks)

- **Modeling**: 22 marks

  1.4 Apply Logistic Regression and LDA (linear discriminant analysis). (4 marks)

  1.5 Apply KNN Model and Naïve Bayes Model. Interpret the results. (4 marks)

  1.6 Model Tuning, Bagging (Random Forest should be applied for Bagging), and Boosting. (7 marks)

  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. (7 marks)

- **Inference**: 5 marks

  1.8 Based on these predictions, what are the insights? (5 marks)

Data Ingestion:

1.1 Read the dataset. Do the descriptive statistics and do the null value condition check. Write an inference on it. (4 Marks)

First of all we will import all the necessary libraries like Pandas, Numpy, seaborn, os to set the path,

After setting the path we will import “Election_Data.xlsx” and sheet named as 'Election_Dataset_Two Classes'.

After importing we will read the dataset. By using head and tail command we will see the First 5 rows and last 5 rows of the dataset, we will get basic idea how data actually is.

First 5 rows,

	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

Last 5 rows,

	Unnamed: 0	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
1520	1521	Conservative	67	5	3	2	4	11	3	male
1521	1522	Conservative	73	2	2	4	4	8	2	male
1522	1523	Labour	37	3	3	5	4	2	2	male
1523	1524	Conservative	61	3	3	1	4	11	2	male
1524	1525	Conservative	74	2	3	2	4	11	0	female

We can see there are 10 columns but the first column we will not consider as this is redundant or of no use, we will remove it further.

Total 9 columns we can see. All seems valid and having significance.

Here All 8 features are Independent variable whereas 9th Feature vote is dependent variable.it is basically Target variable model wants to predict. All features except vote and gender are Numerical. Vote and gender are Nominal Categorical.

Decsription of Data

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
Unnamed: 0	1525	NaN	NaN	NaN	763	440.374	1	382	763	1144	1525
vote	1525	2	Labour	1063	NaN	NaN	NaN	NaN	NaN	NaN	NaN
age	1525	NaN	NaN	NaN	54.1823	15.7112	24	41	53	67	93
economic.cond.national	1525	NaN	NaN	NaN	3.2459	0.880969	1	3	3	4	5
economic.cond.household	1525	NaN	NaN	NaN	3.14033	0.929951	1	3	3	4	5
Blair	1525	NaN	NaN	NaN	3.33443	1.17482	1	2	4	4	5
Hague	1525	NaN	NaN	NaN	2.74689	1.2307	1	2	2	4	5
Europe	1525	NaN	NaN	NaN	6.72852	3.29754	1	4	6	10	11
political.knowledge	1525	NaN	NaN	NaN	1.5423	1.08331	0	0	2	2	3
gender	1525	2	female	812	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In above description we can see mean values and 50% are almost same, mean and median are almost Coherent.

Gender and vote seems to be Categorical Nominal variables, where order is not important aspect

All other variables are Categorical Ordinal Variables, Ratings

All features seems to be somewhat equally distributed around mean.

Information of Data:

```
<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
```

We can see gender and vote are of Object Datatype, we will try to convert it into integer Data type further.

Total 1525 datapoints are there, 1525 different people, No null value can be detected here.

Null value check:

```
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
```

No Null values can be seen in dataset.

Lets drop the column Unnamed: 0, which has no significance here.

Check for Duplicates in dataset:

Number of duplicate rows = 8

We will check manually whether these are exact duplicates or partial duplicates.

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
67	Labour	35	4	4	5	2	3	2	male
626	Labour	39	3	4	4	2	5	2	male
870	Labour	38	2	4	2	2	4	3	male
983	Conservative	74	4	3	2	4	8	2	female
1154	Conservative	53	3	4	2	2	6	0	female
1236	Labour	36	3	3	2	2	6	2	female
1244	Labour	29	4	4	4	2	2	2	female
1438	Labour	40	4	3	4	2	2	2	male

These are the 8 rows which are duplicates. Actually this is not even 1% data we can remove it, but if we can see the age is different for each entry hence this can be data of different personalities. We will keep as it is.

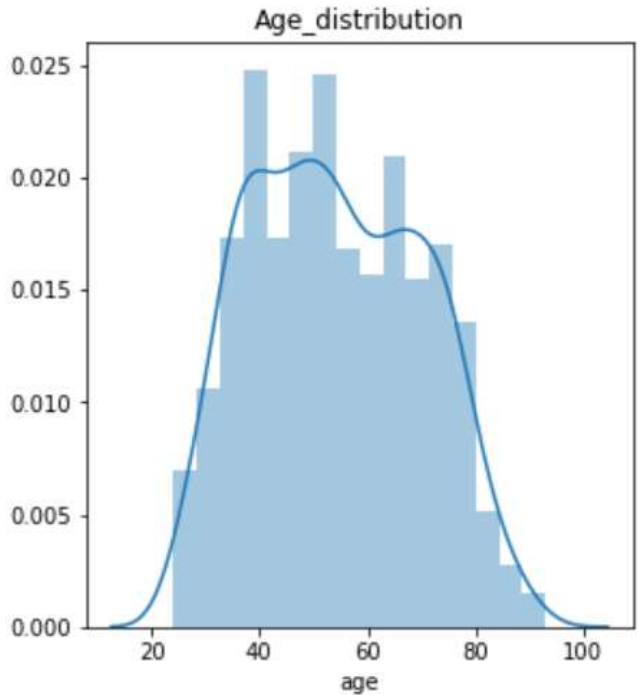
Shape of data:

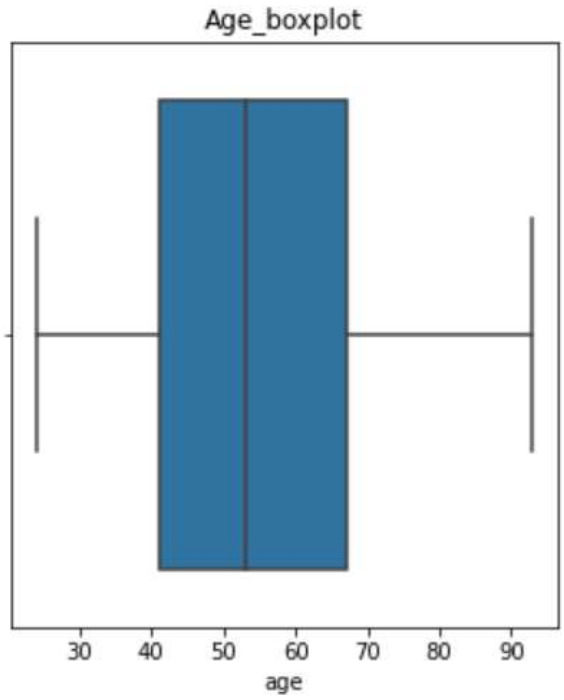
Shape of the dataset is 1525 rows and 9 Columns.

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

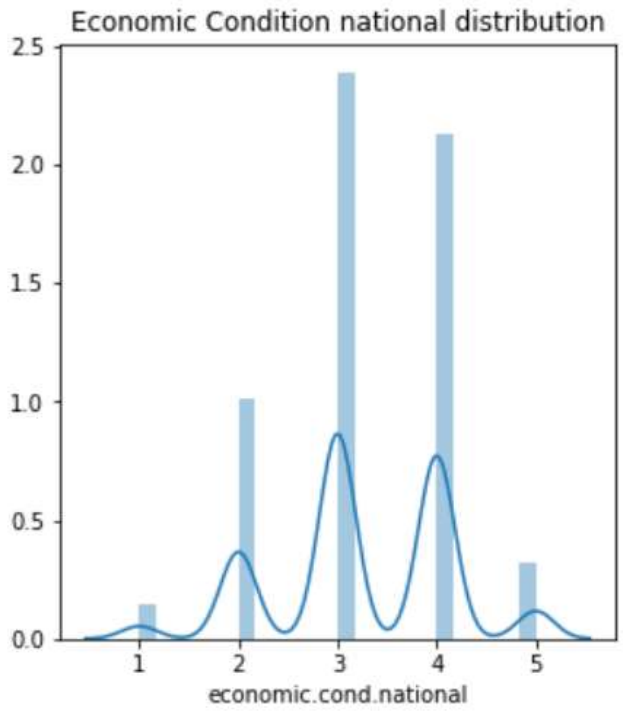
Univariate Analysis:

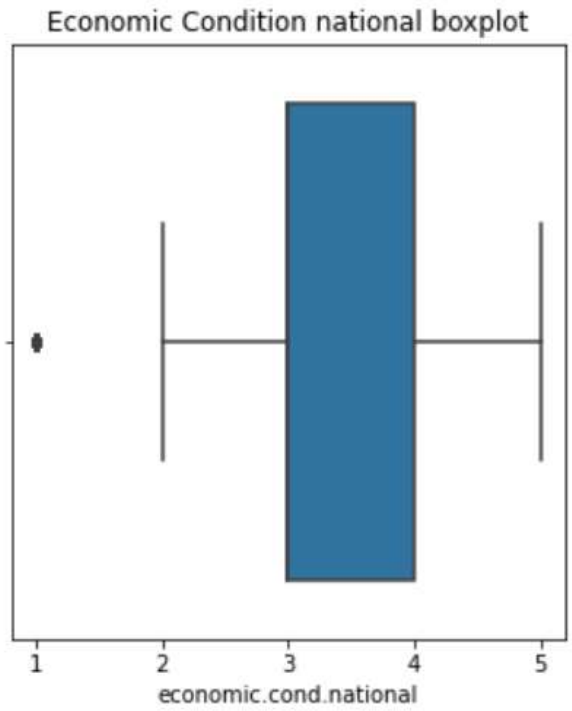
Lets do the univariate analysis, we will check it for each feature separately to get its distribution or any hidden insights.



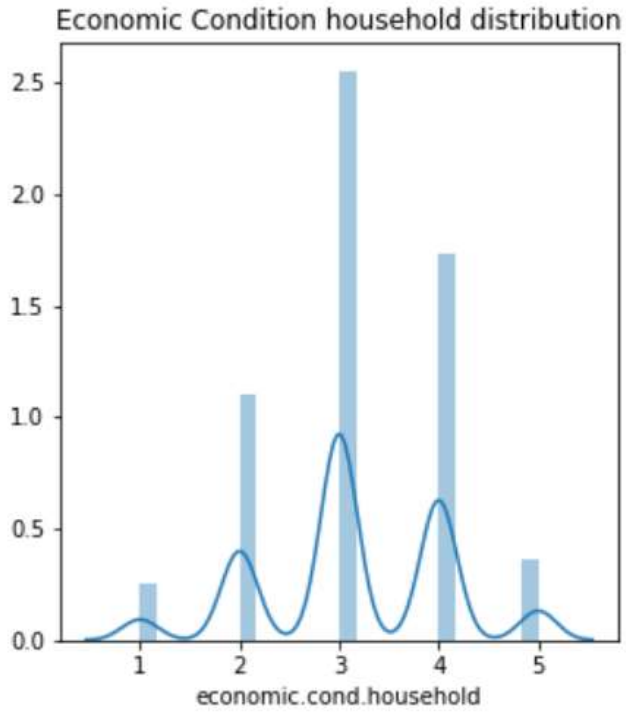


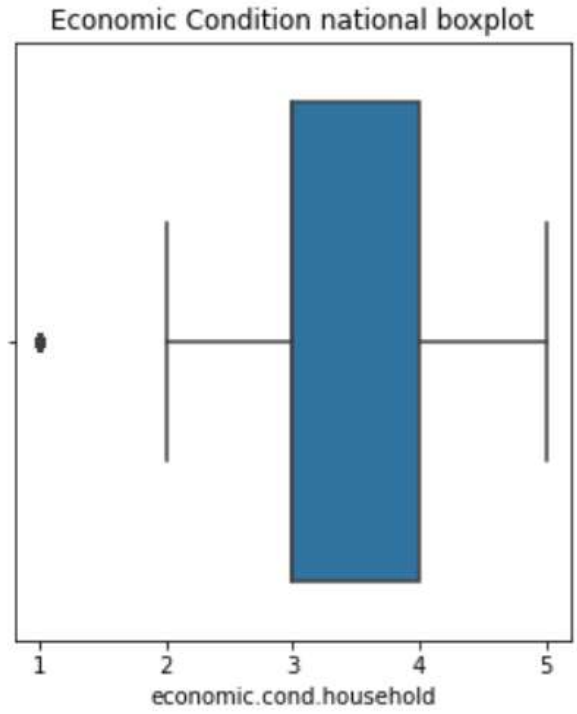
It seems that Age is normally distibuted and not much skewed, all age groups are covered, it has no outliers as well.



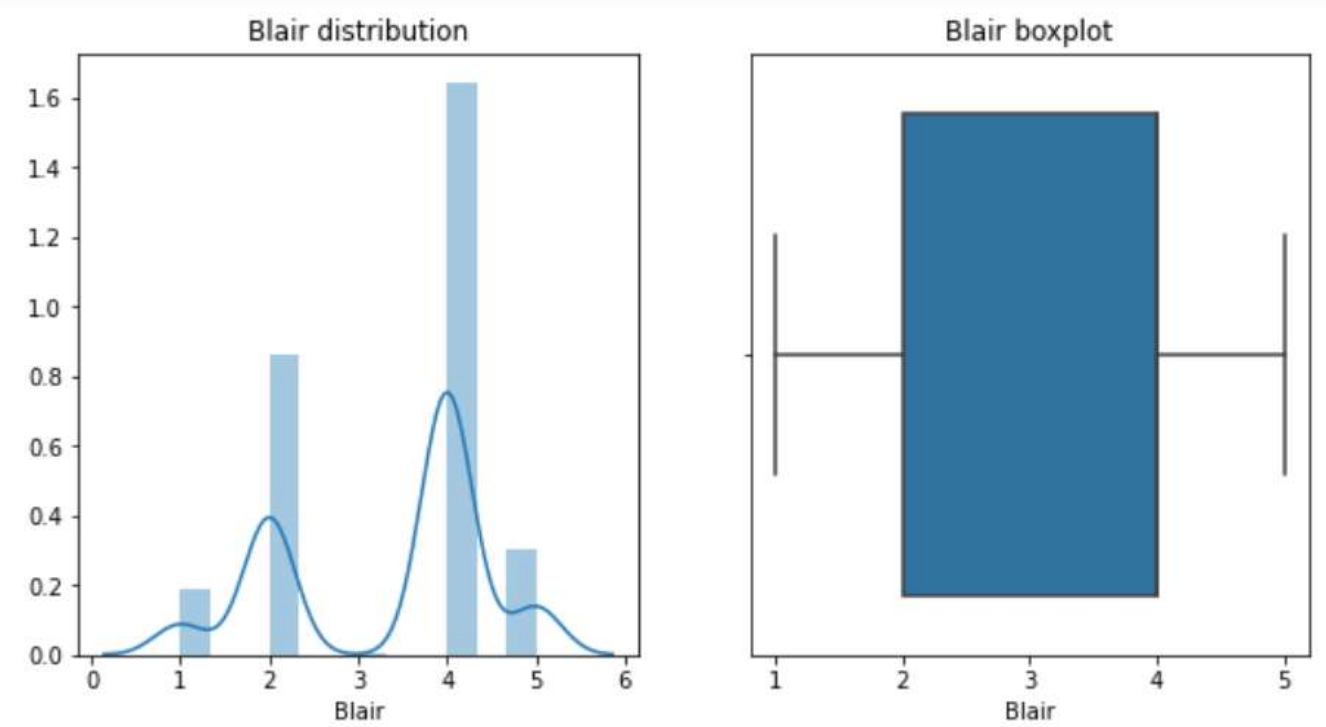


In above plot we can see spikes at almost every rating, but even this is showing most people has given 3 ratings.very less people have given 1 and 5 rating, from this we can say that this particular nation neither have great economical condition nor poor condition.

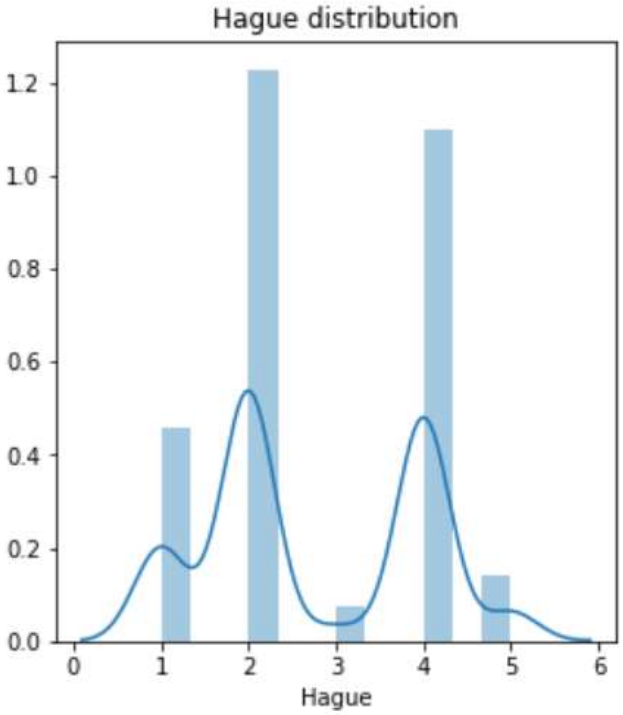


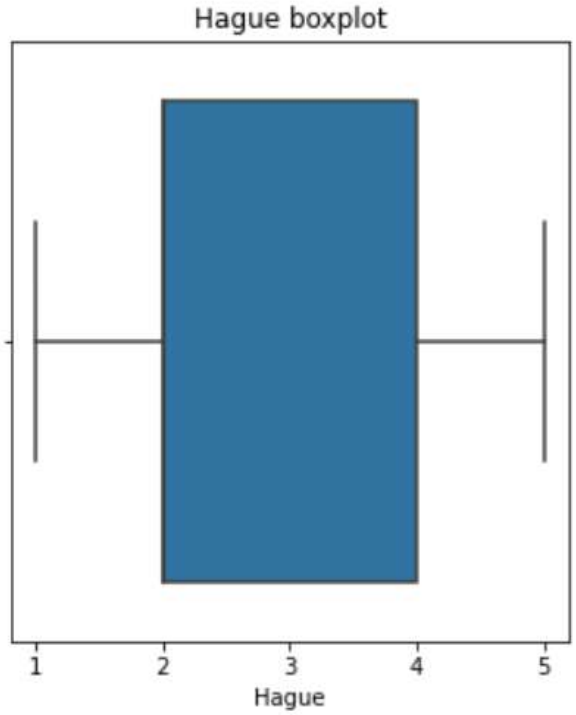


In above plot we can see spikes at almost every rating, but even this is showing most people has given 3 ratings.very less people have given 1 and 5 rating, from this we can say that this particular nation's people neither have great economical condition nor poor condition.

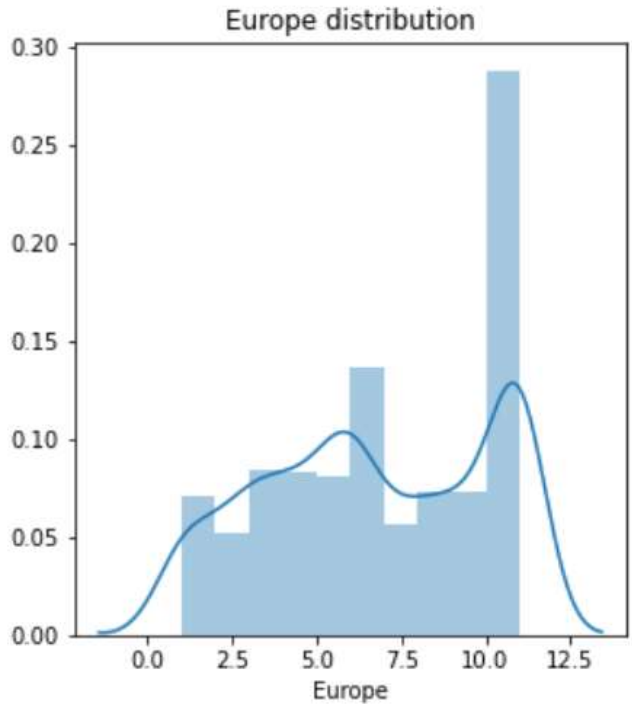


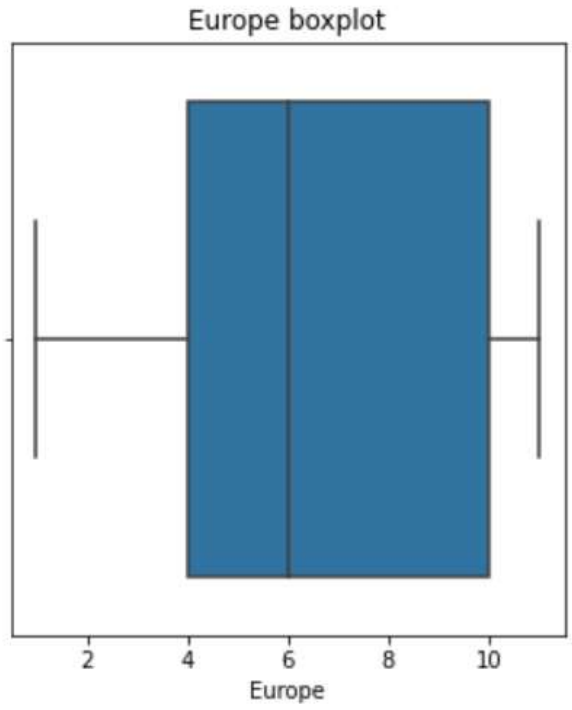
Labour leader Blair has received 2 and 4 score mostly.4 is the highest frequency.



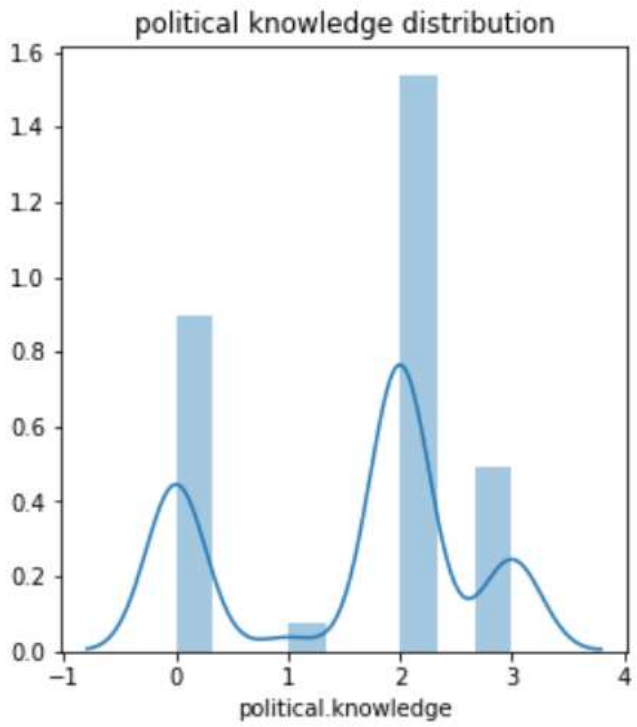


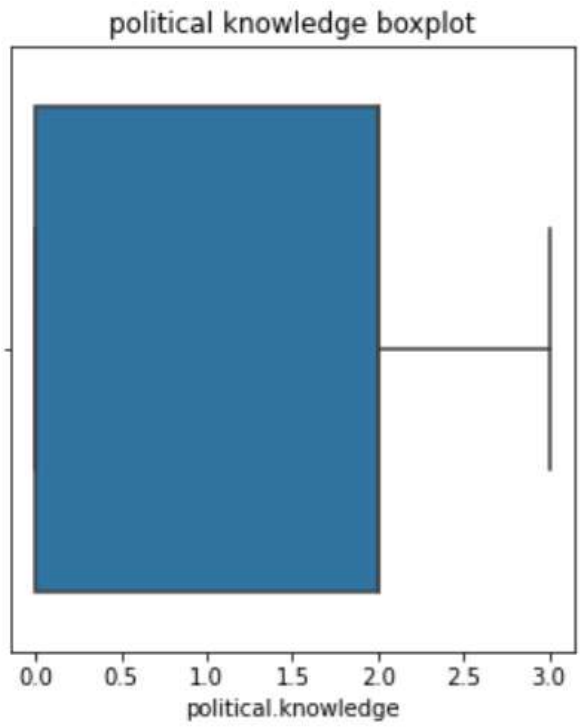
Conservative Party leader Hague has received 2 and 4 score mostly.2 is the highest frequency.





from the Europe plot, the 50% people has rated 4-10 it means they have Eurosceptic sentiment in increasing order.
10 score seems to be have Highest Frequency here, it says that most of the people are against Europe Integration.
Almost 25% people has given rating 10 and 11, they posses Eurosceptic sentiment.
1-4 is the rating given by 25% people, they seems to be with European Integration.





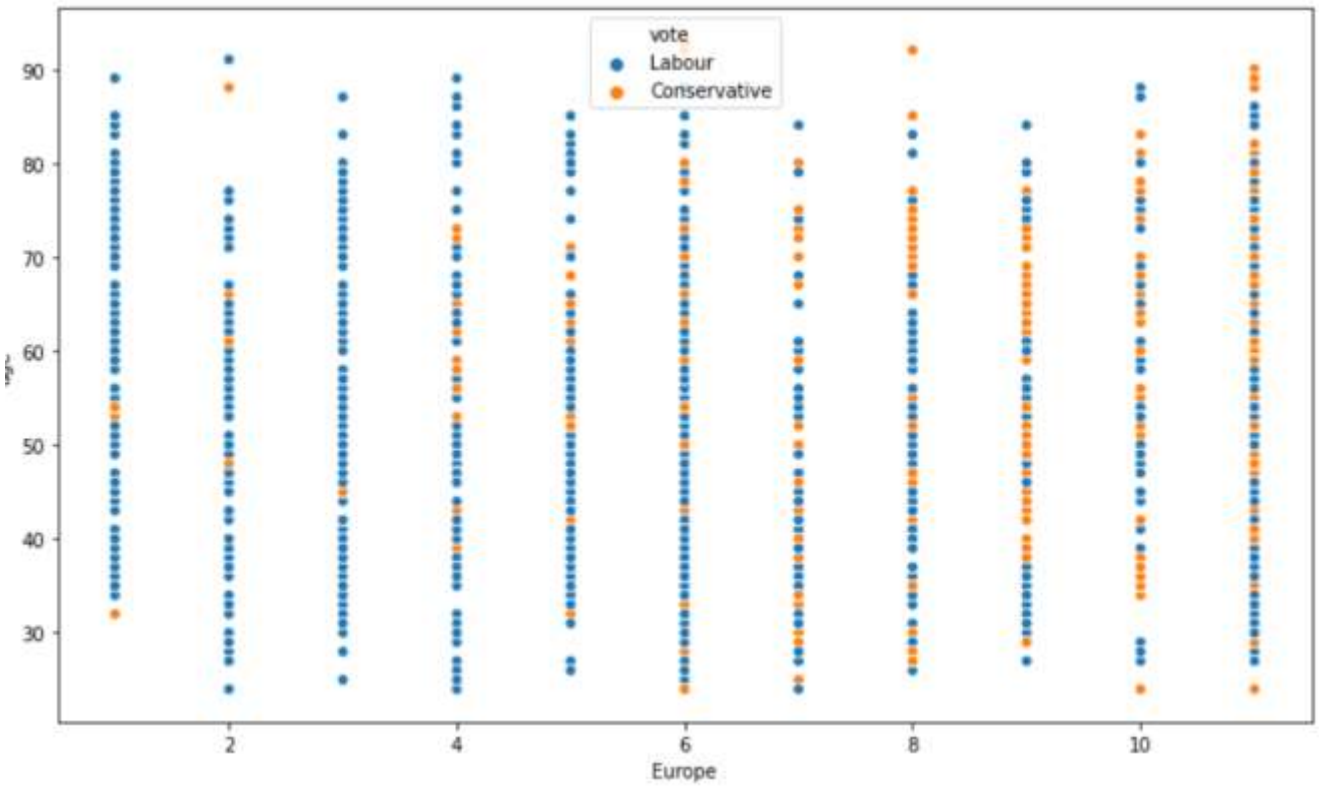
Only around 25% People have average to High political knowledge, 75% people have less Political Knowledge 0,1,2.

Outliers :

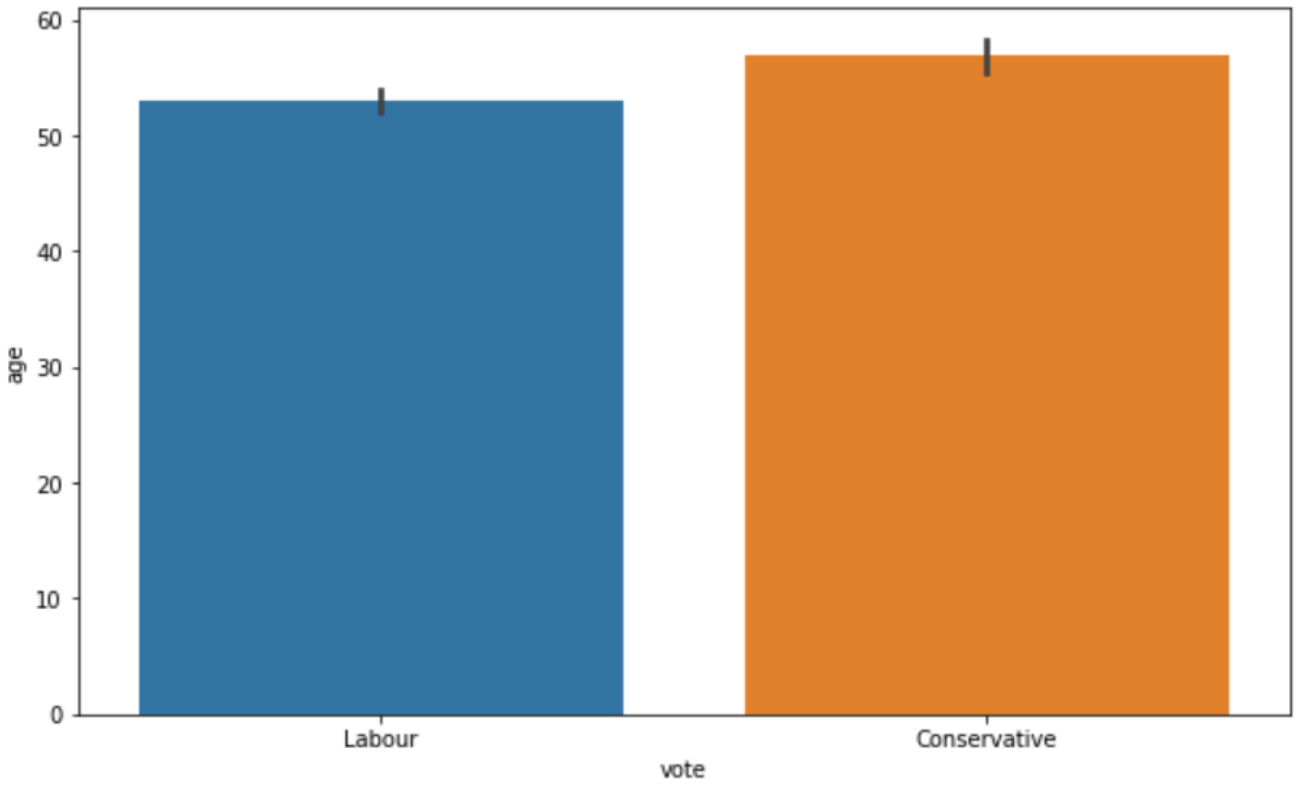
Economic Household conditions and National Household conditions, Rating 1 is outlier as very less number of people has given it. But we will not treat this value as these are valid outliers, we will keep it.

Bivariate Analysis

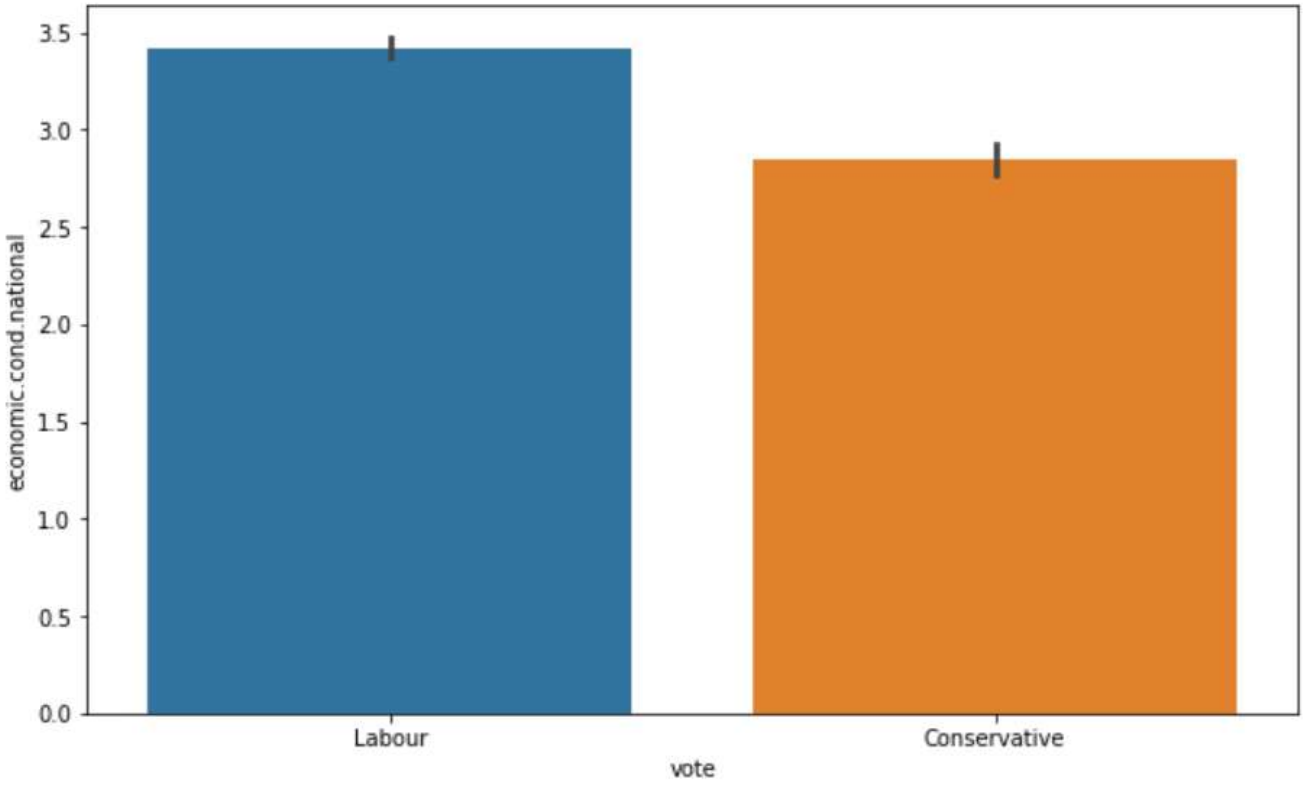
Scatterplot of Europe vs Age



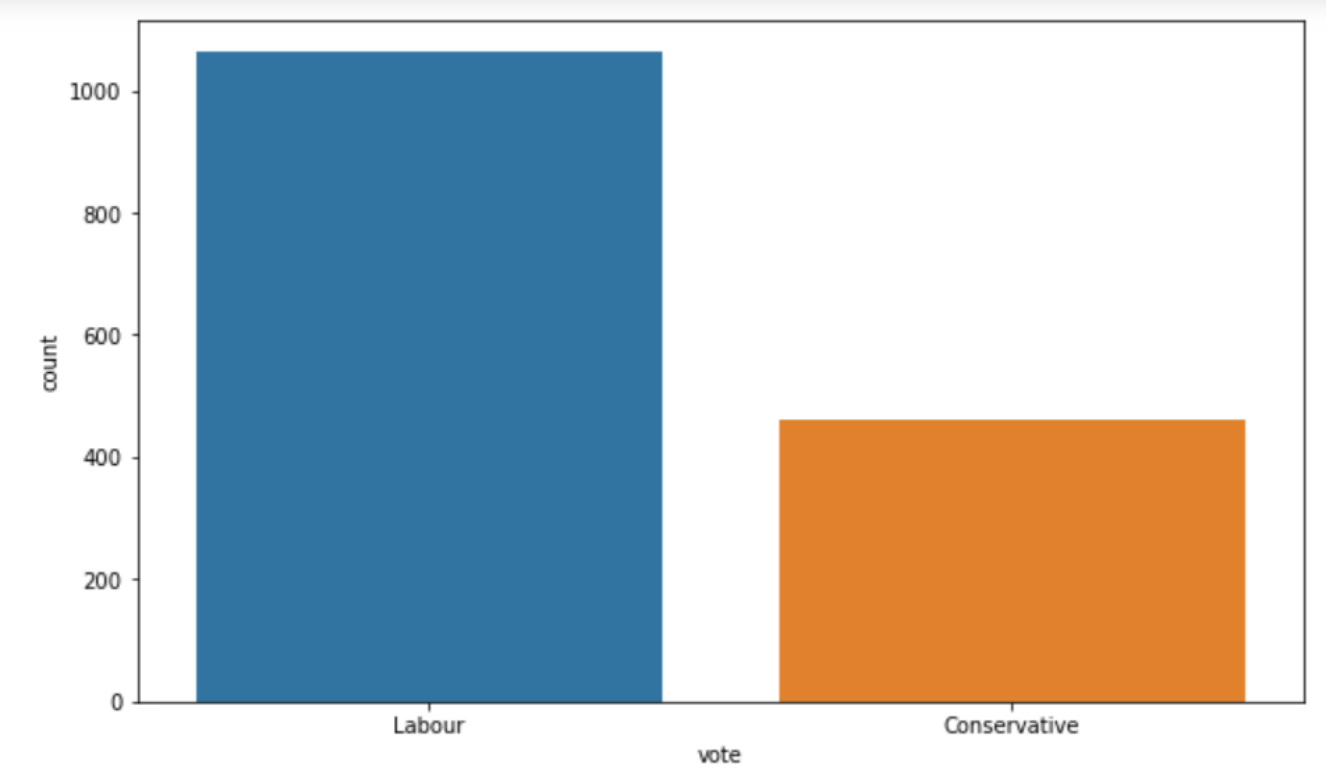
Higher Europe (Eurosceptic) score, it has converted into vote to the Conservative Party. All age group people are voting to both the parties, so it is no more age specific, But Euro Skepticism is playing important role here



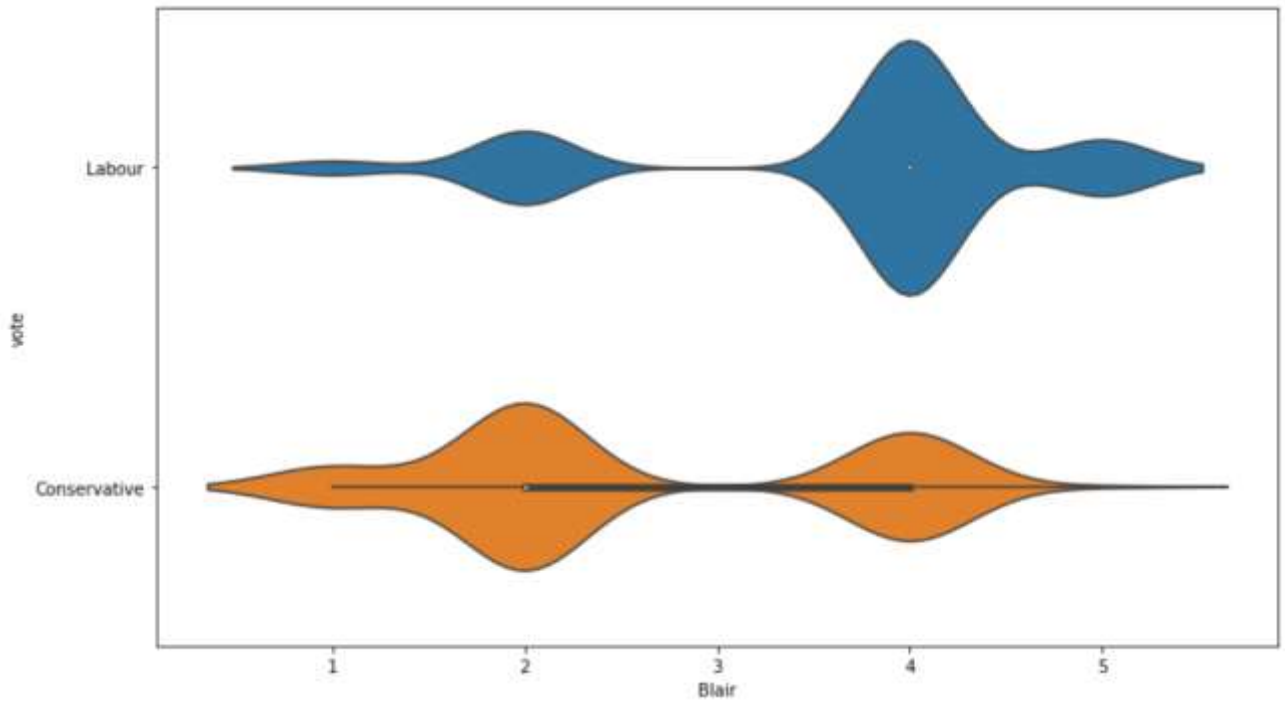
Almost all age groups are voting to Labour and Conservative Party.



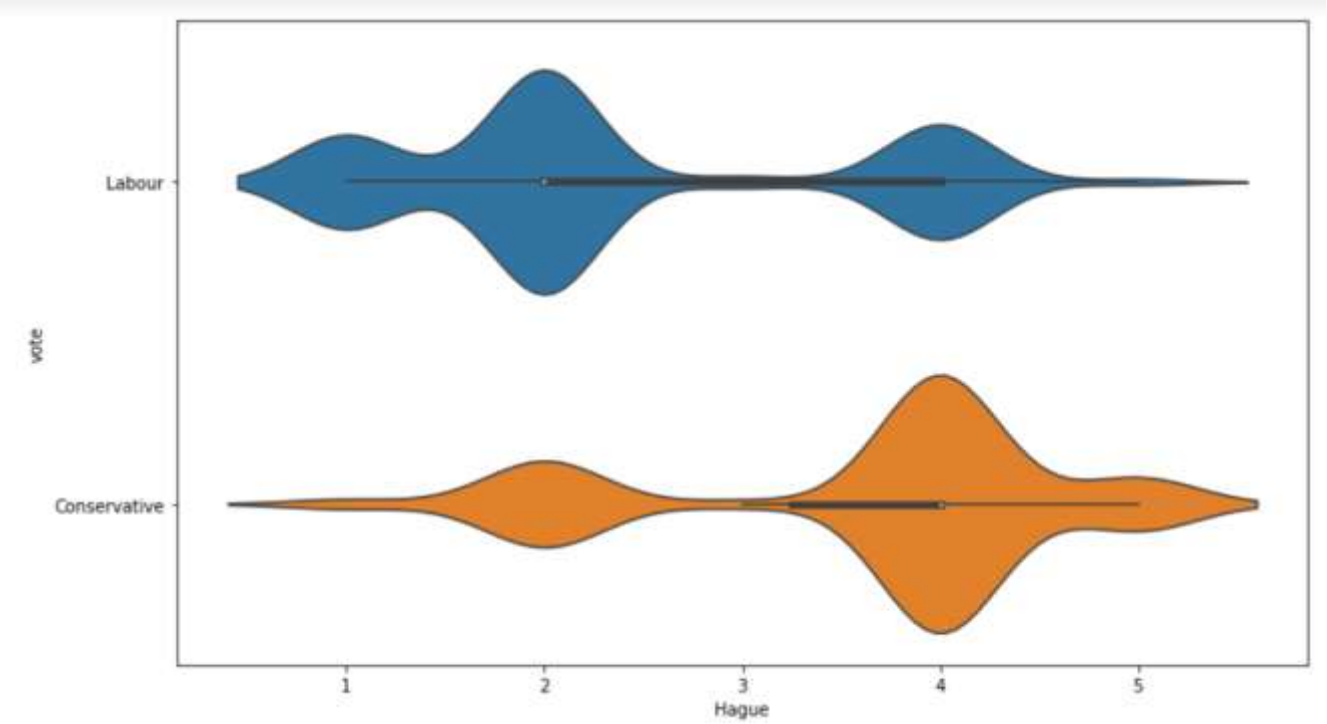
People who are giving good assessment score on National Economic condition are tend to give vote to Labour party, while lesser score turn into vote to Conservative party, this is not the case always happens.



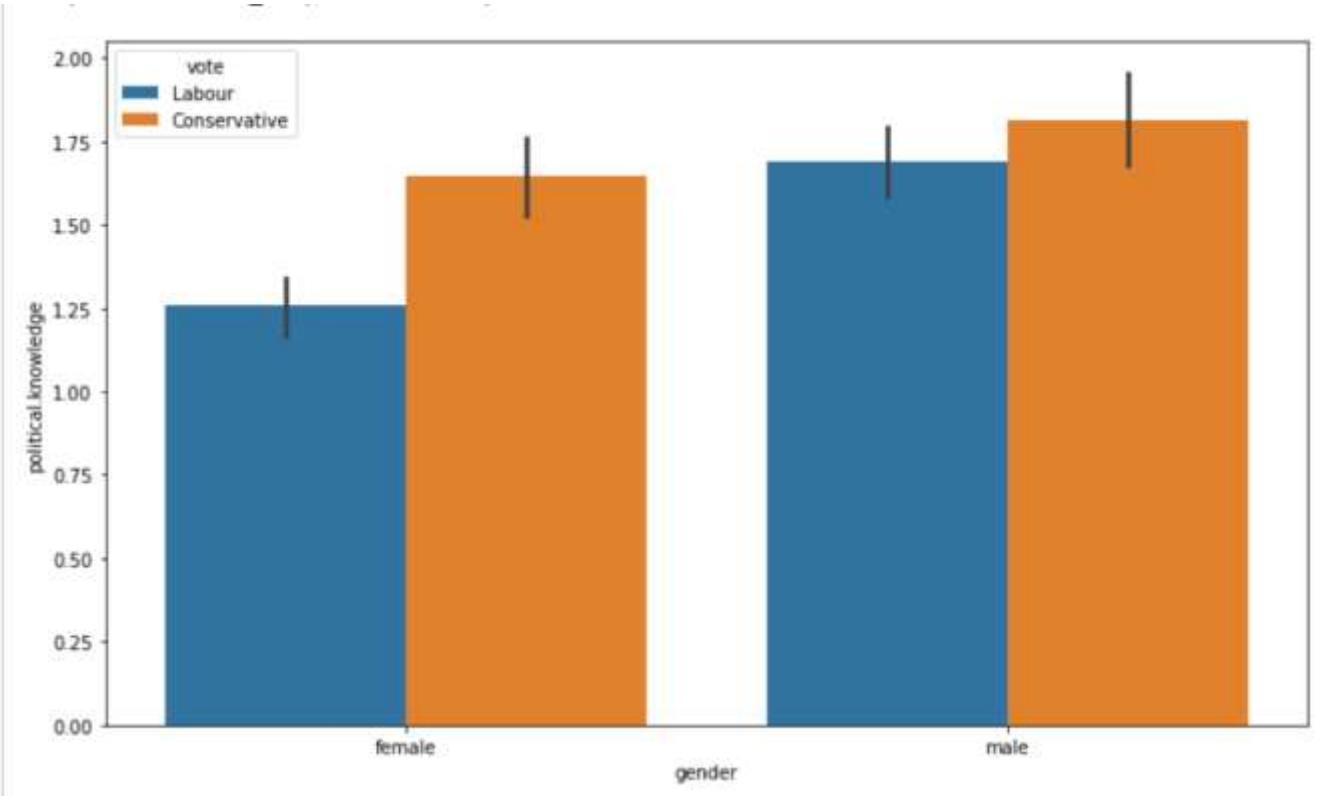
This data has more than 1000 readings who are voting to Labour party while less than 500 who are voting to conservative party it is unbalanced data, might affect to prediction of model.



If person's rating is high to Blair as a party leader, vote will be mostly go into Labour Party's basket. while low Blair score convert the vote into Conservative Party Favor.



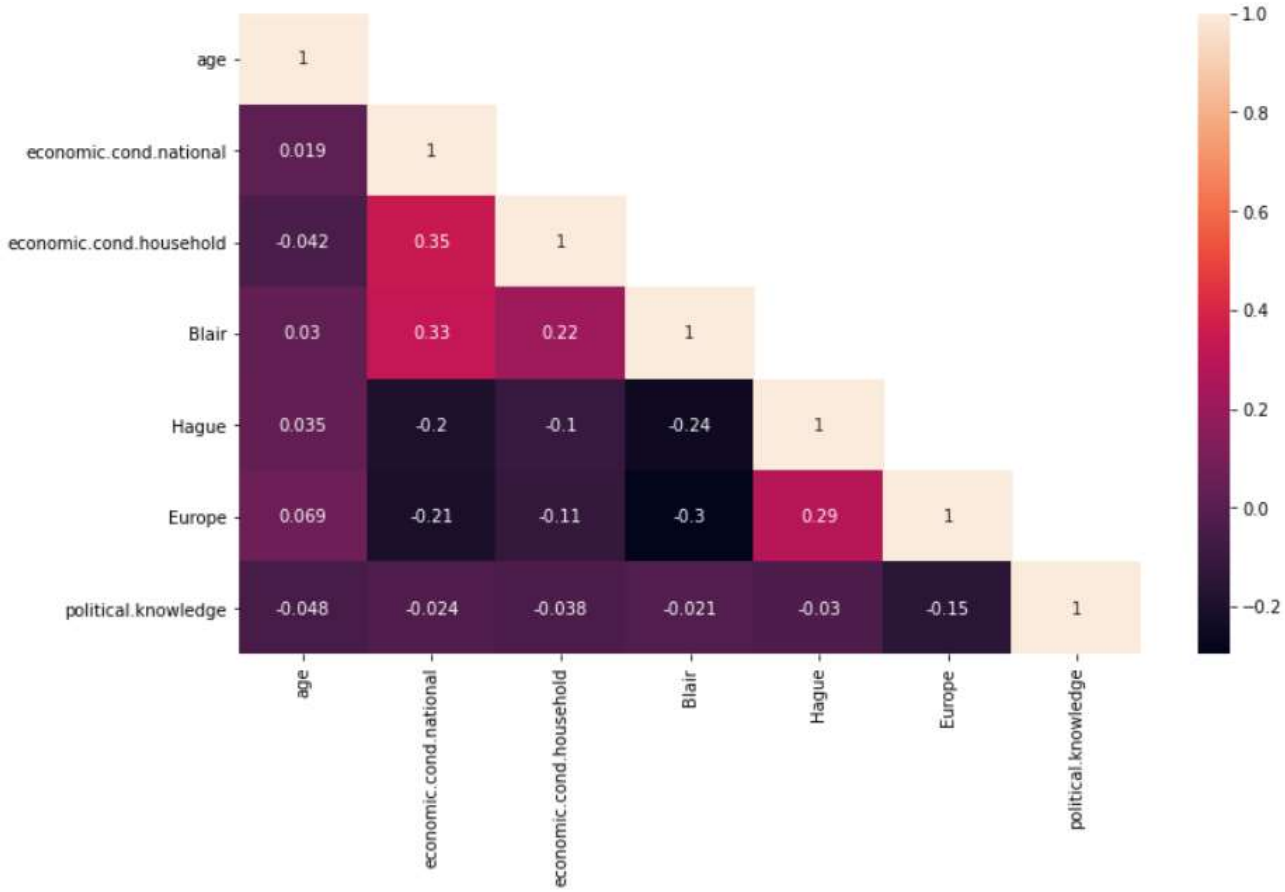
If person's rating is high to Hague as a party leader, vote will be mostly go into Conservative Party's basket. while low Hague score convert the vote into Labour party Favor.



political Knowledge of Males seems to be higher than average knowledge of females. In both cases Higher political Knowledge people are voting to Conservative Party.

Pairplot :

Heatmap for Correlation:



Blair and Economic Household Condition Rating, Economic national Condition rating shows good correlation.

Europe and Blair are inversely related, means if Person is more Eurosceptic there will be less chances he will vote to Blair as a party leader of Labour party

If person is giving good assessment score to Hague he must be with Conservative party and he or she is Highly Eurosceptic.

It is not always true, but people having good political Knowledge are preferring Europe Integration.

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).

We have to encode the features vote and Gender, to feed the data to model.

Here vote and gender are Nominal Categorical variables, we can use one hot encoding to feed data to model.

```
df_data=pd.get_dummies(data=df, drop_first=True)
```

We will use get dummies function to convert data into 0's and 1's.

vote_labour=1 ; Vote has been given to Labour Party.

vote_labour=0 ; Vote has been given to Conservative Party.

gender_male=0; Female

gender_male=1; male

After Scaling, the first 5 rows,

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

All features are having different ranges and different scales like somewhere it is 1-11, 0-4 etc.

We will do scaling for KNN only.

Algorithm	Problem Type	Features might need scaling?
KNN	Either	Yes
Linear regression	Regression	No (unless regularized)
Logistic regression	Classification	No (unless regularized)
Naive Bayes	Classification	No
Decision trees	Either	No
Random Forests	Either	No
AdaBoost	Either	No
Neural networks	Either	Yes

Data Split

Split the data into train and test (70:30).

We will calculate vif to see if there any issue of Multicollinearity or not.

```
age VIF =  1.03
economic.cond.national VIF =  1.28
economic.cond.household VIF =  1.16
Blair VIF =  1.34
Hague VIF =  1.32
Europe VIF =  1.28
political.knowledge VIF =  1.09
vote_Labour VIF =  1.67
gender_male VIF =  1.03
```

All Vif score is less than 4 hence no issue of Multi-collinearity.
Lets copy all the predictor variables into X dataframe. And copy target into the y dataframe.

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

We have created 4 variables here, X_train, X_test, y_train, y_test.

We will check the distribution percentage for target variable in train and test data.

Train target variable:

```
1  y_train.value_counts(1)
```

```
1    0.697282
0    0.302718
Name: vote_Labour, dtype: float64
```

Test Target variable:

```
1  y_test.value_counts(1)
```

```
1    0.696507
0    0.303493
Name: vote_Labour, dtype: float64
```

Almost 70-30 is the distribution of vote is achieved, as original dataset is also contains 70:30 Labour: Conservative distribution.

Modeling:

1.4 Apply Logistic Regression and LDA (linear discriminant analysis). (4 marks)

Apply Logistic Regression :

First we will create Logistic Regression and then we will fit it on train data **X_train, y_train.**

We will use solver as 'newton-cg' with 10,000 iterations.

```
LogisticRegression(max_iter=10000, n_jobs=2, penalty='none', solver='newton-cg',
                   verbose=True)
```

Predicting on Training and Test dataset

```
ytrain_predict = model.predict(X_train)
ytest_predict = model.predict(X_test)
```

Getting the probabilities:

```
ytest_predict_prob=model.predict_proba(X_test)
pd.DataFrame(ytest_predict_prob).head()
```

First 5 rows of Predicted probabilities.

	0	1
0	0.244205	0.755795
1	0.077634	0.922366
2	0.060125	0.939875
3	0.233158	0.766842
4	0.022498	0.977502

Apply Linear Discriminant Analysis:

Build LDA model

We will create LDA classifier and then we will fit it on training data *X_train, y_train.*

```
clf = LinearDiscriminantAnalysis()
model=clf.fit(X_train,y_train)
```

Then we will predict the values on train and test data *X_train , X_test.*
With a cut off value 0.5

1.5 Apply KNN Model and Naïve Bayes Model. Interpret the results.

Apply KNN Model:

Here we have to note one thing is data is not scaled, and KNN algorithm is sensitive to this kind of data, so we will scale the data using *Z-score* before feeding to the model.

We will import zscore from scipy.stats

After applying z score to the data, and after scaling all the dependent variables,

Data is converted from -1 to +1.

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
0	-0.711973	-0.279218	-0.150948	0.566716	-1.419886	-1.434426	0.422643	-0.937059
1	-1.157661	0.856268	0.924730	0.566716	1.018544	-0.524358	0.422643	1.067169
2	-1.221331	0.856268	0.924730	1.418187	-0.607076	-1.131070	0.422643	1.067169
3	-1.921698	0.856268	-1.226625	-1.136225	-1.419886	-0.827714	-1.424148	-0.937059
4	-0.839313	-1.414704	-1.226625	-1.987695	-1.419886	-0.221002	0.422643	1.067169

From *sklearn.neighbors* we will import *KNeighborsClassifier.*

then we will create KNeighborsClassifier()

then we will fit it on train data x_train and y_train.

Apply Naïve Bayes Model:

From sklearn.naive_bayes we will import GaussianNB

Then we will create Naïve Bayes model and fit it on (X_train, y_train).

```
1 NB_model = GaussianNB()
2 NB_model.fit(X_train, y_train)
```

```
GaussianNB()
```

1.6 Model Tuning, Bagging (Random Forest should be applied for Bagging), and Boosting. (7 marks)

Apply Random Forest:

For Bagging we will use Random Forest algorithm as Suggested.

First we will import RandomForestClassifier from sklearn.ensemble.

Then we will create Random forest Classifier with n_estimators as 100.

n_estimators : This is the number of trees we want to build before taking the maximum voting of predictions. Higher number of trees gives better performance.

Then we will fit the model on (X_train, y_train) dataset.

```
from sklearn.ensemble import AdaBoostClassifier

ADB_model = AdaBoostClassifier(n_estimators=100,random_state=1)
ADB_model.fit(X_train,y_train)
```

Apply Bagging

Here we will create Bagging Classifier, Where we will select trees as 100, and base Estimator as Classification and Regression tree.

Then we will fit the model on train data

```
from sklearn.ensemble import BaggingClassifier
Bagging_model=BaggingClassifier(base_estimator=cart,n_estimators=100,random_state=1)
Bagging_model.fit(X_train, y_train)
```

```
BaggingClassifier(base_estimator=DecisionTreeClassifier(), n_estimators=100,
                  random_state=1)
```

Apply Ada boost

First we will import AdaBoostClassifier from sklearn.ensemble library.

Then we will create Ada boost model.

Then we will fit the model on (X_train, y_train) dataset.

```
1  from sklearn.ensemble import AdaBoostClassifier
2
3  ADB_model = AdaBoostClassifier(n_estimators=100,random_state=1)
4  ADB_model.fit(X_train,y_train)
```

```
AdaBoostClassifier(n_estimators=100, random_state=1)
```

Note: Model Tunning is Discussed after Question no. 7

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. (7 marks)

So far we have created and trained the models Logistic Regression, LDA, Naive Bayes, KNN, Random Forest, Ada boost on Training data set. Now we will check Performance of all of them.

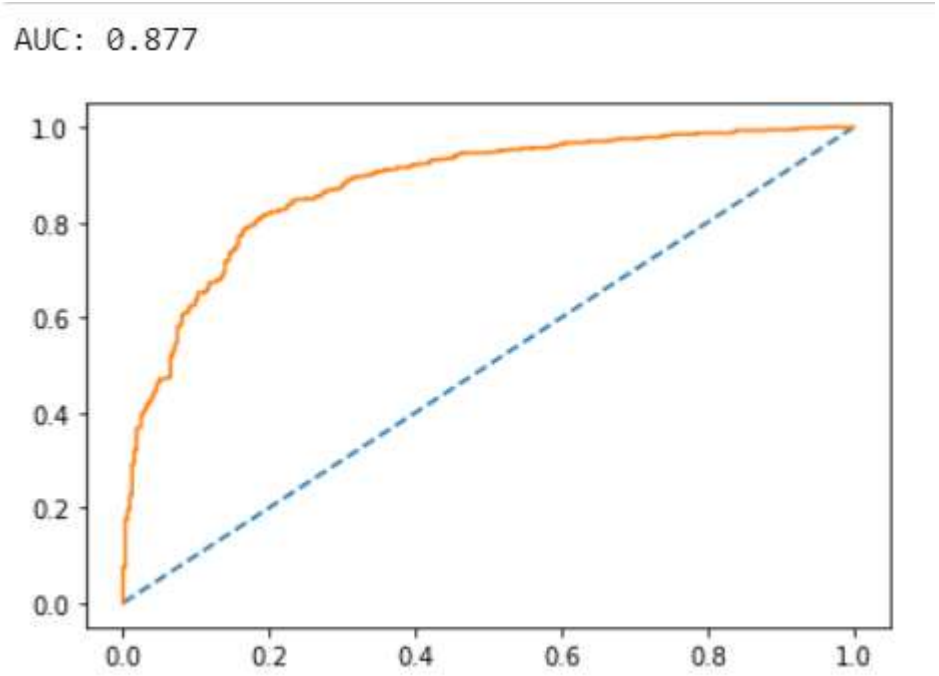
1. Logistic Regression

Accuracy and AUC-ROC:

On Train data:

83% is the Accuracy on train data.

AUC and ROC curve for training data:

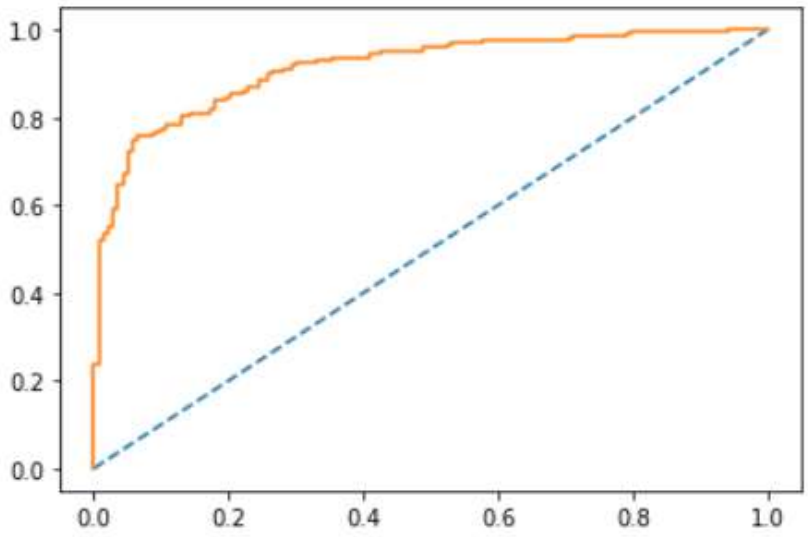


On Test data:

84.93% is the Accuracy on train data.

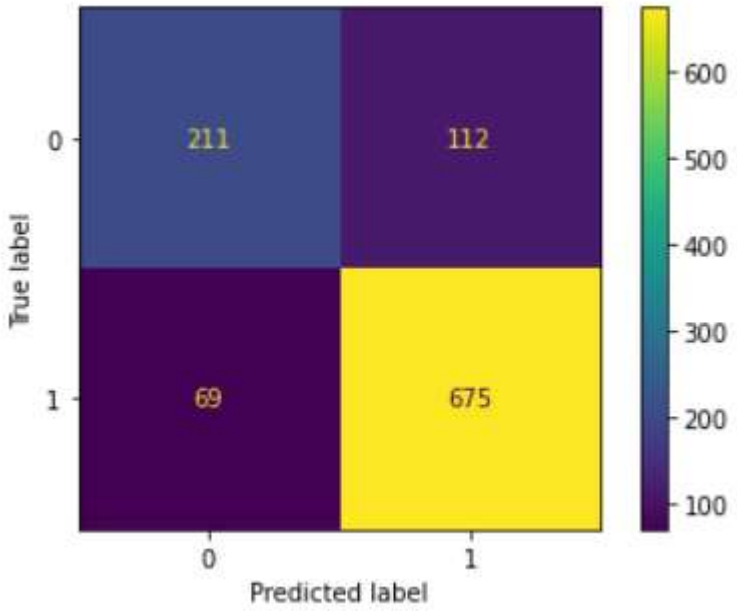
AUC and ROC curve for training data:

AUC: 0.914

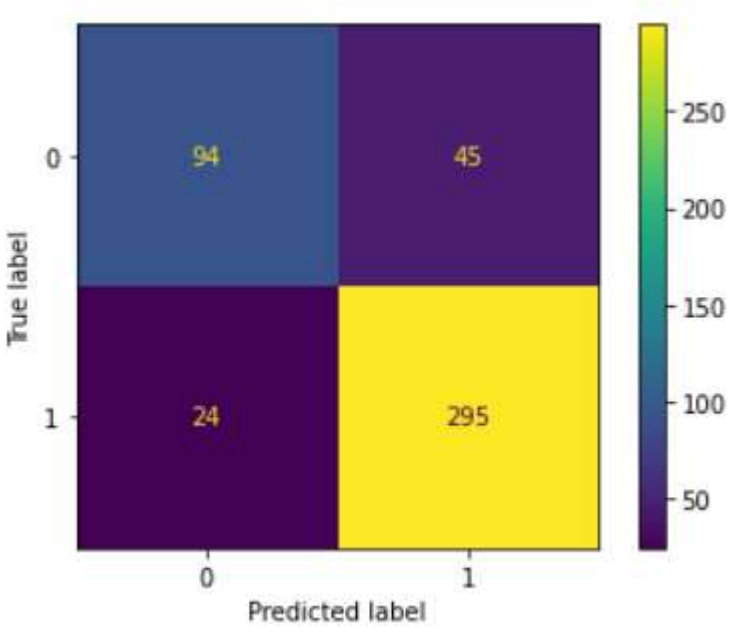


Confusion Matrix

On Train data



On Test Data : Confusion Matrix,



Classification Report

On Train data:

	precision	recall	f1-score	support
0	0.75	0.65	0.70	323
1	0.86	0.91	0.88	744
accuracy			0.83	1067
macro avg	0.81	0.78	0.79	1067
weighted avg	0.83	0.83	0.83	1067

```
lr_train_precision  0.86
lr_train_recall  0.91
lr_train_f1  0.88
```

On Test Data ,

	precision	recall	f1-score	support
0	0.80	0.68	0.73	139
1	0.87	0.92	0.90	319
accuracy			0.85	458
macro avg	0.83	0.80	0.81	458
weighted avg	0.85	0.85	0.85	458

```
lr_test_precision  0.87
lr_test_recall  0.92
lr_test_f1  0.9
```

We can see both 0's and 1's are equally Important here, Because vote to both leaders matter to us. Logistic Regression has performed really well no overfitting issue can be Observed here.

Precision, recall and f1 score to predict 0, is 75%, 65% , 70% on Train data.

It has 83%-84% Accuracy on test data, while f1 score is also 88-90%.

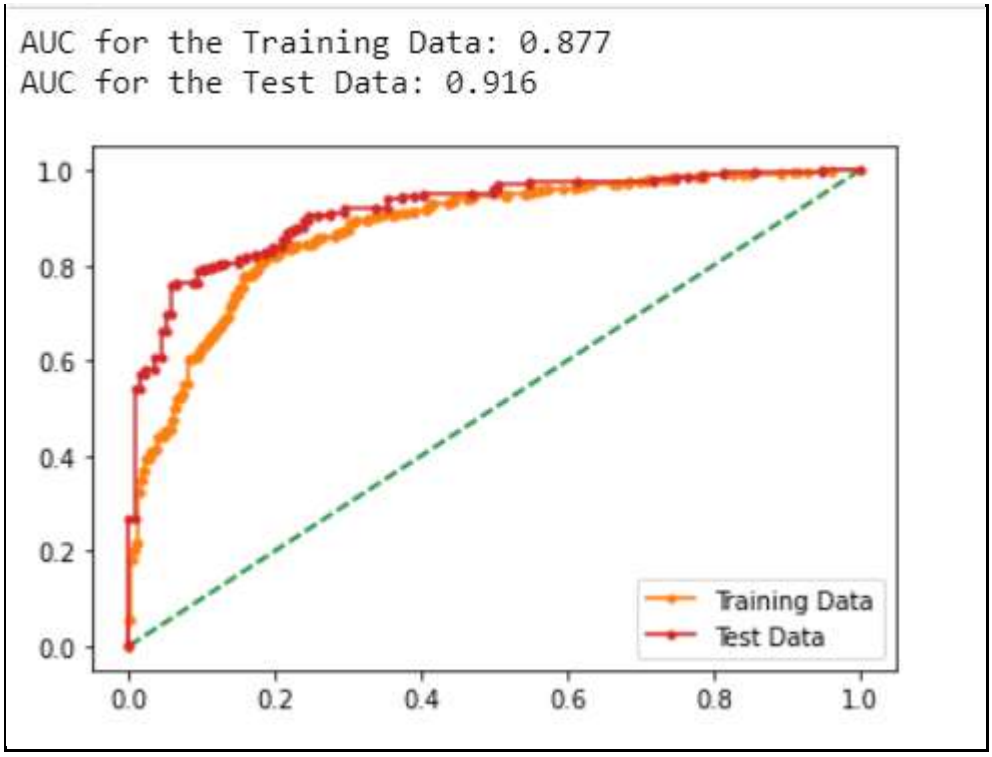
2. Linear Discriminant Analysis

Accuracy and AUC-ROC:

On Train and Test data:

83% and 85% is the Accuracy on train data and test data respectively.

AUC and ROC curve for training and test data:



Confusion Matrix for LDA:

Classification Report for LDA:

Classification Report of the training data:

	precision	recall	f1-score	support
0	0.74	0.67	0.70	323
1	0.86	0.90	0.88	744
accuracy			0.83	1067
macro avg	0.80	0.78	0.79	1067
weighted avg	0.83	0.83	0.83	1067

Classification Report of the test data:

	precision	recall	f1-score	support
0	0.79	0.71	0.75	139
1	0.88	0.92	0.90	319
accuracy			0.85	458
macro avg	0.83	0.81	0.82	458
weighted avg	0.85	0.85	0.85	458

LDA has also performed really well no overfitting issue can be Observed here. On Test data and train data only + - 2% is the difference.

To Predict 0, the precision, recall and f1 score has fallen as compare to Predict 1.

It has 83%-85% Accuracy, while f1 score is also 88-90%.

3. K Nearest Neighbour

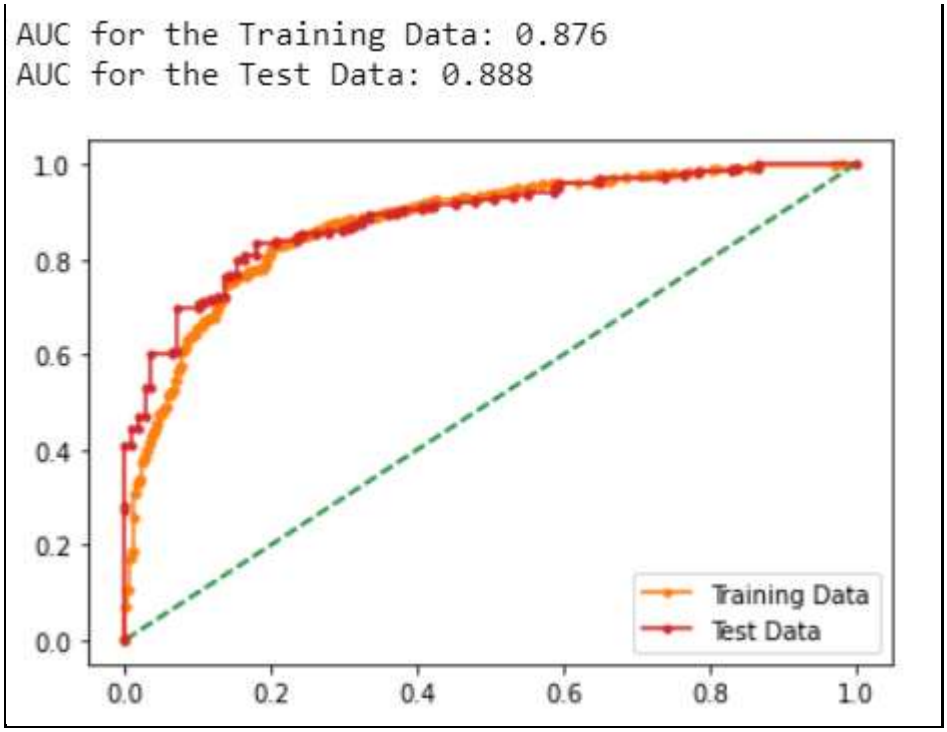
Accuracy and AUC-ROC:

On Train data:

87% is the Accuracy on train data.

82% is the Accuracy on test data.

AUC and ROC curve for and test data:



Confusion Matrix for KNN:

Classification Report for KNN :

Classification report for training data:

	precision	recall	f1-score	support
0	0.81	0.75	0.78	351
1	0.89	0.92	0.91	792
accuracy			0.87	1143
macro avg	0.85	0.83	0.84	1143
weighted avg	0.87	0.87	0.87	1143

Classification report for test data:

	precision	recall	f1-score	support
0	0.69	0.73	0.71	111
1	0.89	0.86	0.87	271
accuracy			0.82	382
macro avg	0.79	0.80	0.79	382
weighted avg	0.83	0.82	0.83	382

Here in KNN we can see The value is little lower in case of Test Data

Accuracy and f1 score, but again even this model will perform well,

We need to Hyperparameter tuning to improve performance.

It is Predicting 1 with Higher f1 score but it is showing poor performance to Predict 0.

4. Naive Bayes Algorithm

Accuracy and AUC-ROC:

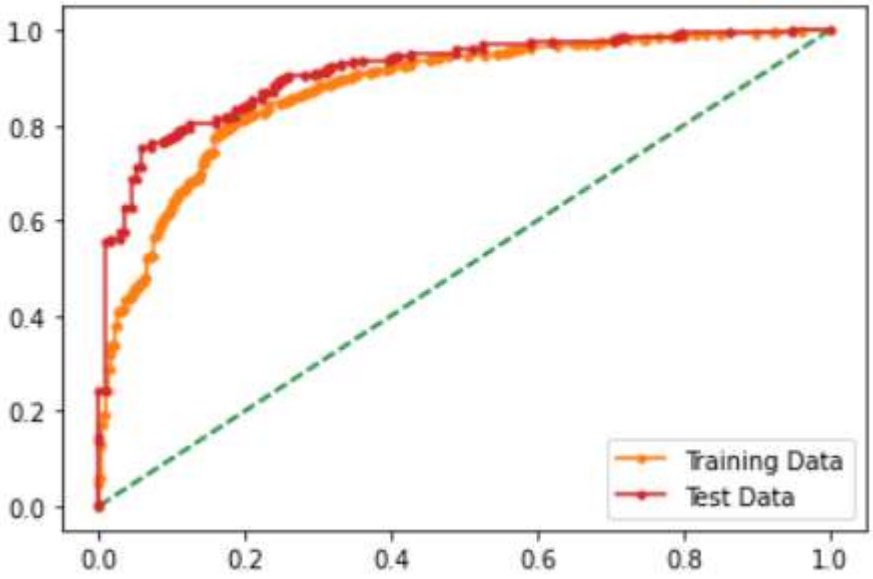
On Train data:

82% is the Accuracy on train data.

84% is the Accuracy on test data.

AUC and ROC curve for train and test data:

```
AUC for the Training Data: 0.876
AUC for the Test Data: 0.915
```



Confusion Matrix :

Classification Report for Naïve Bayes Algorithm:

On Train data:

```
0.8219306466729147
[[223 100]
 [ 90 654]]
              precision    recall  f1-score   support

           0       0.71      0.69      0.70       323
           1       0.87      0.88      0.87       744

    accuracy                           0.82      1067
   macro avg       0.79      0.78      0.79      1067
weighted avg       0.82      0.82      0.82      1067
```

On Test Data:

```
0.8471615720524017
[[101  38]
 [ 32 287]]
              precision    recall  f1-score   support

           0       0.76      0.73      0.74       139
           1       0.88      0.90      0.89       319

    accuracy                           0.85       458
   macro avg       0.82      0.81      0.82       458
weighted avg       0.85      0.85      0.85       458
```

We need Hyperparameter tuning to improve performance.

It is Predicting 1 with Higher f1 score but it is showing poor performance to Predict 0.

5. Random Forest

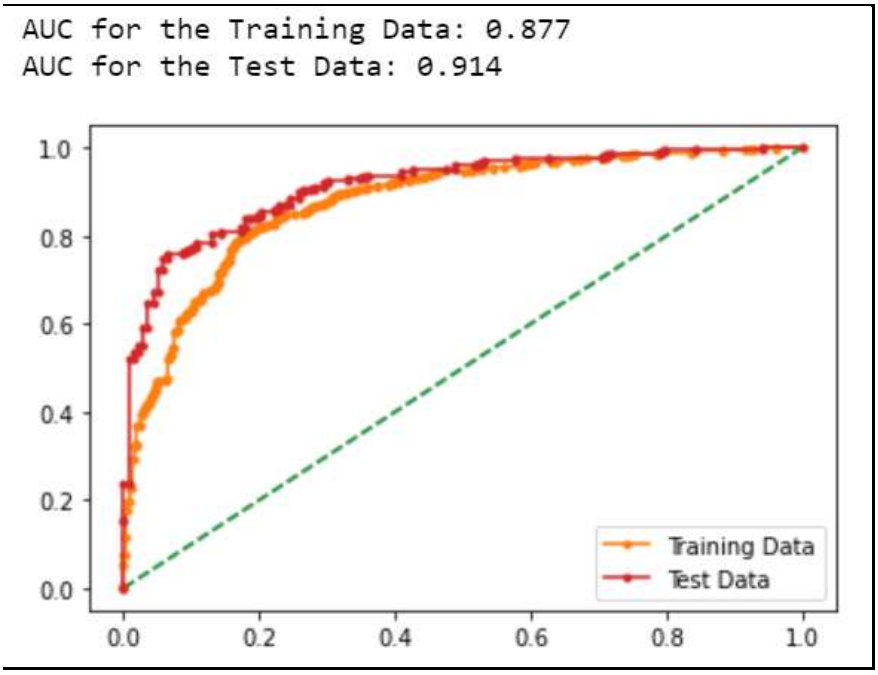
Accuracy and AUC-ROC:

On Train-test data:

100% is the Accuracy on train data.

85% is the Accuracy on test data.

AUC and ROC curve for train and test data:



Confusion Matrix for Random Forest :

Classification Report for Random Forest:

On Train Data:

```
0.9990627928772259
[[322   1]
 [  0 744]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       323
           1       1.00      1.00      1.00       744

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

On Test Data:

```
0.8493449781659389
[[ 94  45]
 [ 24 295]]
              precision    recall  f1-score   support

           0       0.80      0.68      0.73       139
           1       0.87      0.92      0.90       319

    accuracy                           0.85       458
   macro avg       0.83      0.80      0.81       458
weighted avg       0.85      0.85      0.85       458
```

Random Forest is Clearly a overfitting one, as almost all metrics has reduced on test data.

6. Bagging

Accuracy and AUC-ROC:

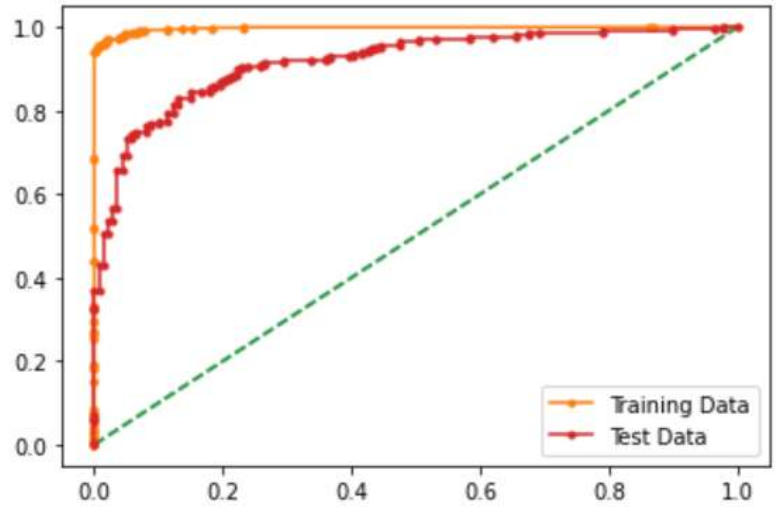
On Train-test data:

99.7% is the Accuracy on train data.

91.8% is the Accuracy on test data.

AUC and ROC curve for train and test data:

```
AUC for the Training Data: 0.997
AUC for the Test Data: 0.918
```



Confusion Matrix for Bagging :

Classification Report for Bagging:

On Train Data,

```
0.971883786316776
[[298  25]
 [  5 739]]
              precision    recall  f1-score   support

           0       0.98      0.92      0.95       323
           1       0.97      0.99      0.98       744

    accuracy                           0.97      1067
   macro avg       0.98      0.96      0.97      1067
weighted avg       0.97      0.97      0.97      1067
```

On Test data,

```
0.8427947598253275
[[ 93  46]
 [ 26 293]]
              precision    recall  f1-score   support

           0       0.78      0.67      0.72       139
           1       0.86      0.92      0.89       319

    accuracy                           0.84       458
   macro avg       0.82      0.79      0.81       458
weighted avg       0.84      0.84      0.84       458
```

Model has stability in predicting 1 while Predicting 0, Bagging Model is Overfitting.

7. Ada Boost

Accuracy and AUC-ROC:

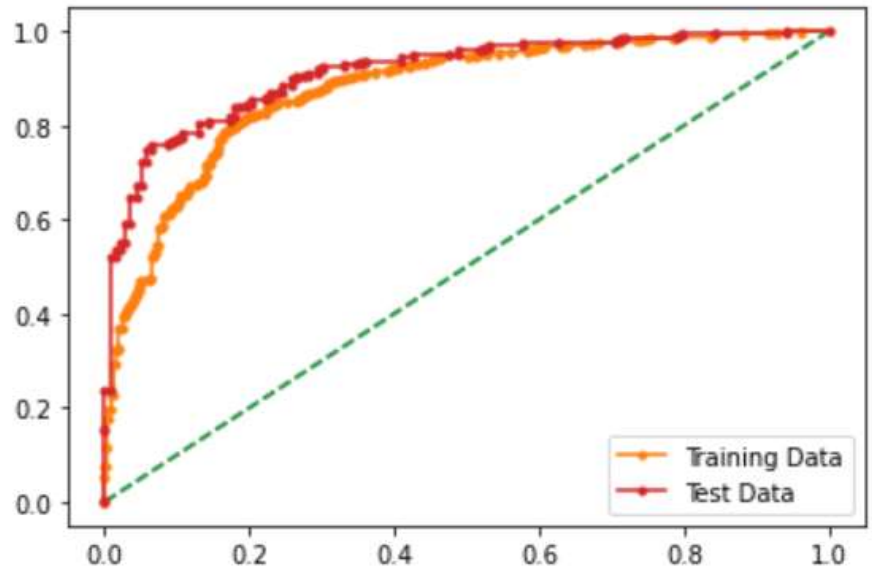
On Train-test data:

84.5% is the Accuracy on train data.

83.62% is the Accuracy on test data.

AUC and ROC curve for train and test data:

```
AUC for the Training Data: 0.877
AUC for the Test Data: 0.914
```



Confusion Matrix for Ada Boost:

Classification Report for Ada Boost:

On Train Data,

```
0.8444236176194939
[[227  96]
 [ 70 674]]
              precision    recall  f1-score   support

           0       0.76      0.70      0.73       323
           1       0.88      0.91      0.89       744

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

On Test Data,

```
0.8362445414847162
[[ 94  45]
 [ 30 289]]
              precision    recall  f1-score   support

           0       0.76      0.68      0.71       139
           1       0.87      0.91      0.89       319

    accuracy                           0.84       458
   macro avg       0.81      0.79      0.80       458
weighted avg       0.83      0.84      0.83       458
```

Model is Performing well as compare to other models.

All Model Comparison by its performance metrics:

	Logit Train	Logit Test	LDA Train	LDA Test	KNN Train	KNN Test	Naive Bayes Train	Naive Bayes Test	RF Train	RF Test	Ada Boost Train	Ada Boost Test	Ada Boost Train_0	Ada Boost Test_0
Accuracy	0.83	0.85	0.83	0.84	0.87	0.71	0.82	0.85	1.00	0.85	0.84	0.84	0.84	0.84
AUC	0.88	0.91	0.88	0.91	0.88	0.89	0.88	0.91	0.88	0.91	0.90	0.91	0.90	0.91
Recall	0.91	0.92	0.90	0.91	0.92	0.86	0.88	0.90	1.00	0.92	0.70	0.91	0.70	0.68
Precision	0.86	0.87	0.86	0.87	0.89	0.89	0.87	0.88	1.00	0.87	0.76	0.87	0.76	0.76
F1 Score	0.88	0.90	0.88	0.89	0.91	0.87	0.87	0.89	1.00	0.90	0.73	0.89	0.73	0.71

Now the Train data seems to be unbalanced, like 30-70% Conservative Party and Labour party votes, so we will use SMOTE oversampling method on train data here to balance the Data.

Naive Bayes with SMOTE

Performance metrics for Naive Bayes with SMOTE:

On Train Data :

```
0.8131720430107527
[[595 149]
 [129 615]]
              precision    recall  f1-score   support

           0       0.82      0.80      0.81       744
           1       0.80      0.83      0.82       744

    accuracy                           0.81      1488
   macro avg       0.81      0.81      0.81      1488
weighted avg       0.81      0.81      0.81      1488
```

On Test Data:

```
0.8427947598253275
[[109  30]
 [ 42 277]]
              precision    recall  f1-score   support

           0       0.72      0.78      0.75       139
           1       0.90      0.87      0.88       319

    accuracy                           0.84       458
   macro avg       0.81      0.83      0.82       458
weighted avg       0.85      0.84      0.84       458
```

Model is overfitting here, so we will see what happens with SMOTE and Ada Boost, Linear Regression.

SMOTE with Ada Boost:

Classification Report:

On Train Data,

```
0.8145161290322581
[[601 143]
 [133 611]]
              precision    recall  f1-score   support

           0       0.82      0.81      0.81       744
           1       0.81      0.82      0.82       744

    accuracy                           0.81      1488
   macro avg       0.81      0.81      0.81      1488
weighted avg       0.81      0.81      0.81      1488
```

On Test Data,

```
0.8362445414847162
[[110  29]
 [ 46 273]]
              precision    recall  f1-score   support

           0       0.71      0.79      0.75       139
           1       0.90      0.86      0.88       319

    accuracy                           0.84       458
   macro avg       0.80      0.82      0.81       458
weighted avg       0.84      0.84      0.84       458
```

This is again Overfitting/Underfitting issue while predicting 0's and 1.

SMOTE with Logistic Regression:

Classification Report:

On Train Data,

```
0.821236559139785
[[611 133]
 [133 611]]
              precision    recall  f1-score   support

           0       0.82      0.82      0.82       744
           1       0.82      0.82      0.82       744

    accuracy                           0.82      1488
   macro avg       0.82      0.82      0.82      1488
weighted avg       0.82      0.82      0.82      1488
```

On Test Data,

```
0.8362445414847162
[[114  25]
 [ 50 269]]
              precision    recall  f1-score   support

           0       0.70      0.82      0.75       139
           1       0.91      0.84      0.88       319

    accuracy                           0.84       458
   macro avg       0.81      0.83      0.82       458
weighted avg       0.85      0.84      0.84       458
```

Not exactly but Logistic Regression seems much stable as Compare to other models, after applying SMOTE-balanced data.

Model Tunning :

Now we will use Model tunning for various Algorithms to check whether the performance is Improving or not by changing its hyperparameters.

For Naive Bayes Model, will use K-fold validation how model performs on Limited dataset.

```
Cross Validation Score: [0.82242991 0.8411215  0.81308411 0.81308411 0.81308411 0.82242991
 0.79439252 0.87735849 0.81132075 0.81132075] [0.82242991 0.8411215  0.81308411 0.81308411 0.81308411 0.82242991
 0.79439252 0.87735849 0.81132075 0.81132075]
Average Score: 0.8219626168224299
```

Average Score is 0.82, which is good score, it is performing well.

Lets check using GRID search, for KNN

```
params = {'n_neighbors':[2,4,6,8,10,12,14,16,18],
          'algorithm': ['auto','ball_tree','kd_tree','brute'],
          'leaf_size':list(range(1,30)),
          'p':[1,2],
          'metric':['minkowski','euclidean','manhattan','chebyshev', 'mahanalobis']}
```

We are here using different distance metrics, P value is

When p = 1, this is equivalent to using manhattan_distance

P=2,Power parameter for the Minkowski metric.

We are selecting leaf size from 1 to 30, to get the optimal value.

Before fitting the data, we are scaling train data by using z score

Scaled data:

age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
0.497751	-0.279218	-0.150948	-1.136225	1.831354	0.385710	-1.424148	1.067169
-0.902983	-0.279218	0.924730	0.566716	-1.419886	-1.131070	1.346038	1.067169
-0.138946	-1.414704	-0.150948	-1.136225	1.831354	1.295778	0.422643	-0.937059
-0.457295	-0.279218	-0.150948	0.566716	-0.607076	-0.827714	-1.424148	-0.937059
-0.202616	-0.279218	-1.226625	0.566716	-0.607076	-1.434426	0.422643	-0.937059
...	...	...	...	...	...	...	...
0.816100	0.856268	-0.150948	1.418187	-0.607076	-1.434426	0.422643	1.067169
-1.730689	1.991754	2.000408	-1.136225	-1.419886	-0.827714	1.346038	1.067169
-1.285001	0.856268	2.000408	0.566716	1.018544	0.082354	0.422643	-0.937059
-1.157661	0.856268	0.924730	0.566716	-0.607076	0.082354	0.422643	-0.937059
-1.412340	-1.414704	-1.226625	0.566716	1.018544	-0.524358	1.346038	1.067169

The values set for algorithm,

```
GridSearchCV(estimator=KNeighborsClassifier(),
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                         'leaf_size': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
                                       13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
                                       23, 24, 25, 26, 27, 28, 29],
                         'metric': ['minkowski', 'euclidean', 'manhattan',
                                    'chebyshev', 'mahanalobis'],
                         'n_neighbors': [2, 4, 6, 8, 10, 12, 14, 16, 18],
                         'p': [1, 2]},
             verbose=1)
```

These are the best Parameters for KNN:

```
{'algorithm': 'auto',
 'leaf_size': 1,
 'metric': 'minkowski',
 'n_neighbors': 16,
 'p': 2}
```

Train and Test Accuracy for KNN after Grid Search CV.

```
Train Accuracy is :0.8433945756780402

Test Accuracy is :0.8272251308900523
```

It is almost same as earlier.

Lets check using GRID search, for Logistic Regression and Random Forest

We will import Pipeline from sklearn.pipeline library.

In this we will create two Classifier – Logistic Regression and Random Forest will see the best performing model with its best Parameters.

```
{'classifier': LogisticRegression(C=11.288378916846883, solver='liblinear'),
 'classifier__C': 11.288378916846883,
 'classifier__penalty': 'l2',
 'classifier__solver': 'liblinear'}
```

Logistic Regression Classiofier is giving best results,with Solver Liblinear
With Classifier Penalty as Ridge.

```
Train Accuracy is :0.8294283036551078

Test Accuracy is :0.8493449781659389
```

Classification Report:

```
Classification report for Train set:
              precision    recall  f1-score   support

           0       0.75      0.65      0.70       323
           1       0.86      0.91      0.88       744

    accuracy                           0.83      1067
   macro avg       0.80      0.78      0.79      1067
weighted avg       0.83      0.83      0.83      1067
```

```
Classification report for Test set:
              precision    recall  f1-score   support

           0       0.80      0.68      0.73       139
           1       0.87      0.92      0.90       319

    accuracy                           0.85       458
   macro avg       0.83      0.80      0.81       458
weighted avg       0.85      0.85      0.85       458
```

The Logistic Regression is Performing well on Test data and Train data, it seems to be stable. It is predicting 0 and 1 clearly.

It has low scores while predicting 0 but for 1 Precision, Recall and F1 is good on both train and test data.

Prediction of 0 i.e vote to conservative Party, prediction is little tougher as there is very less data points conveying vote to Conservative Party. This can be minimized if we ask for more data.

If we check the first 10 votes, It is clearly, mentioning the vote to Labour party.

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1], dtype=uint8)
```

First 400 votes,

```
{0: 96, 1: 304}
```

Here out of 400, 96 votes are to the Conservative party and 304 votes are to the Labour party so clearly it is indicating the Labour party will win.
Again there are some assumptions and data limits we have, so as of now this is the scenario.

1.8 Based on these predictions, what are the insights?

Summary:

In this Case study we have to decide strategy or plan or we have to predict which party will win election, To which party particular person will vote. What are the deciding factors to win election that we are studying here. Survey is Conducted on 1525 different people, where features like Age, whether person is Eurosceptic with ratings, Assessment to both leaders Hague and Blair. Person is having political Knowledge or not. what is public opinion on Household and national economy, their gender is also taken into consideration, so it seems we have sufficient features to evaluate result.

Steps Performed and Insights:

To predict results, we have started with Exploratory data Analysis where we got some hidden insights by looking at Dataset, by using univariate analysis, Multivariate analysis. Null value, Outlier detection. From EDA we have drawn below mentioned Insights:

Assesment to leader Blair and Economic Household Condition Rating, Economic national Condition rating shows good correlation.

Europscepticism score and Blair Assesment score are inversly related, means if Person is more Eurosceptic there will be less chances he will vote to Blair as a party leader of Labour party.

If person is giving good assesment score to Hague he must be with Conservative party and he or she is Highly Eurosceptic.

It is not always true, but people having good political Knowledge are preferring Europe Integration.

Modelling

As the target variable here is Categorcal i.e Conservative and Labour, we can use Logistic Regression, LDA, KNN, Naive Bayes, RF, Bagging, Boosting.

After all this we have created a Final Classification report where all performance metrics are mentioned.

We have used all mentioned Algorithms. If we check the Target Variable we have almost 70% data of Labour Party and 30 % Conservative Party. We have built all algorithms on this data first, after this we have used SMOTE technique of Oversampling to balance Imbalanced data. We can see in the Classification report that Accuracy, Precision has been dropped.

After all this to do Model tunning we have used Gridsearch CV, K fold cross validation to Improve performance of Existing Base models.

Results:

To get the crystal clear picture, we need more data points to avoid Imbalance data issue. but from this data, we can say that to win the election one must be with vision Europe integration, as this vision has helped labour party.

The Model Logistic Regression has good score and seems to be very stable model in this Scenario. If we predict on Test Data, The labour Party seems to be get Higher number of Votes as compare to Conservative Party. Again the question arises here does this sample represent the whole nation, any bias in the data, like from Particular region data is collected where Labour Party has higher influence. There are less votes to Conservative party, but again it needs more data to predict correct results. But as of now this data tells us that Labour party will win this elections.