MACHINE LEARNING PROJECT BUSINESS REPORT

BY

RITUSRI MOHAN

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.

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

The dataset has 1525 rows and 10 columns. The head of the dataset can be seen below. There are 10 columns. The first column, i.e., 'Unnamed: 0' is dropped from the dataset as it does not contribute to further analysis.

	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

From the summary of the dataset below, it is seen that there is difference in the scale of the values across the columns. There is a very small chance for outliers to be present as there is barely any difference between the 75% value and max value for some variables.

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

Presence of null values across the variables was found to be zero as it can be seen below. The data types of variables are also presented. There are 7 variables with integer data type and 2 variables with object data type.

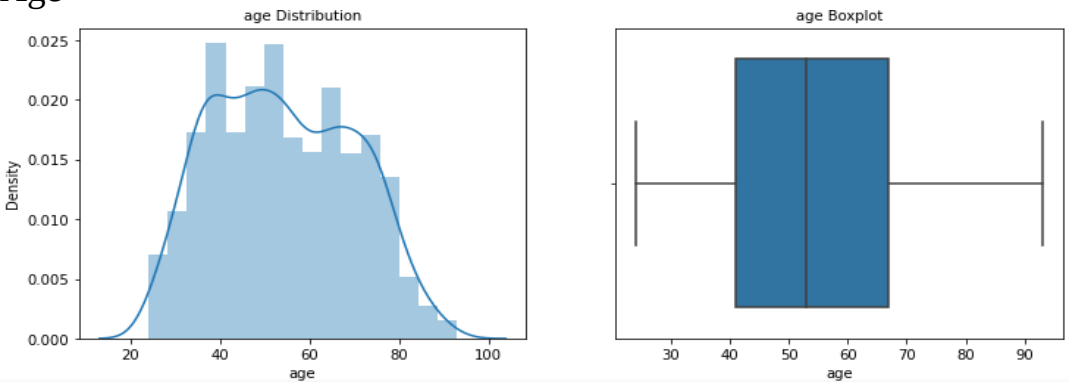
```
vote                      0
age                       0
economic.cond.national    0
economic.cond.household   0
Blair                     0
Hague                     0
Europe                    0
political.knowledge       0
gender                    0
dtype: int64
```

```
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   vote                    1525 non-null   object
 1   age                     1525 non-null   int64
 2   economic.cond.national  1525 non-null   int64
 3   economic.cond.household 1525 non-null   int64
 4   Blair                   1525 non-null   int64
 5   Hague                   1525 non-null   int64
 6   Europe                  1525 non-null   int64
 7   political.knowledge     1525 non-null   int64
 8   gender                  1525 non-null   object
dtypes: int64(7), object(2)
memory usage: 107.4+ KB
```

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

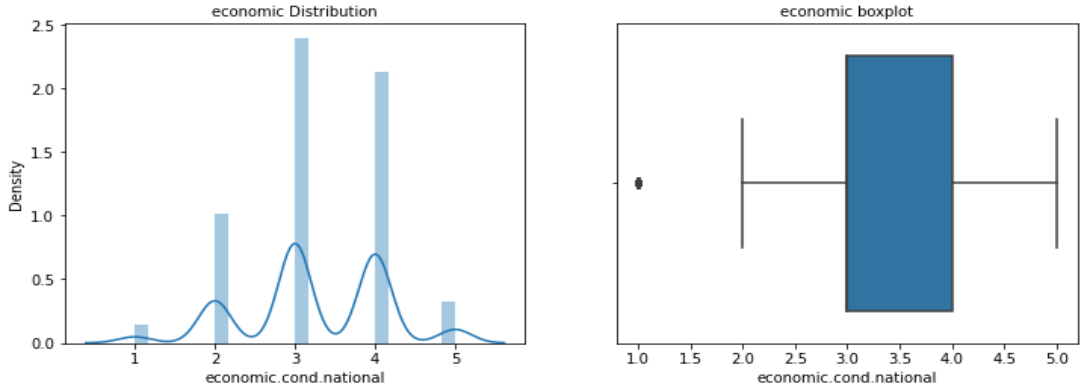
<u>Univariate Analysis</u>

- Age



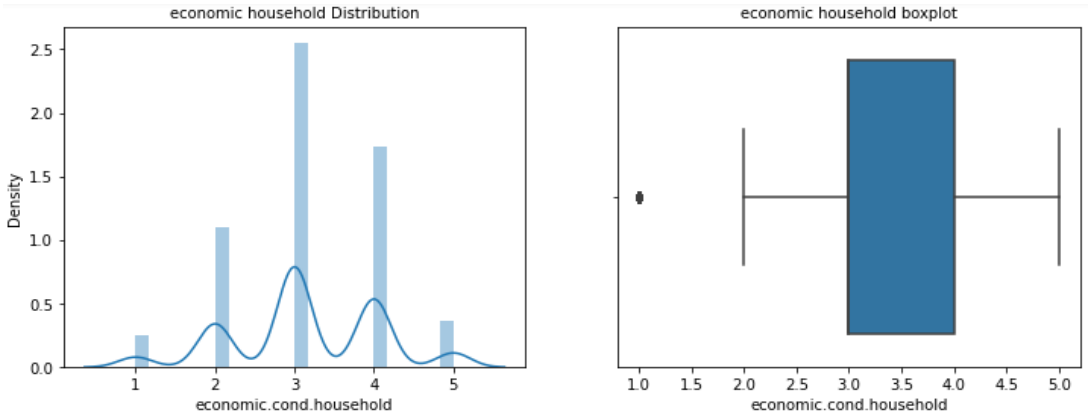
The data points seem to be normally distributed without any outliers.

- Economic.cond.national



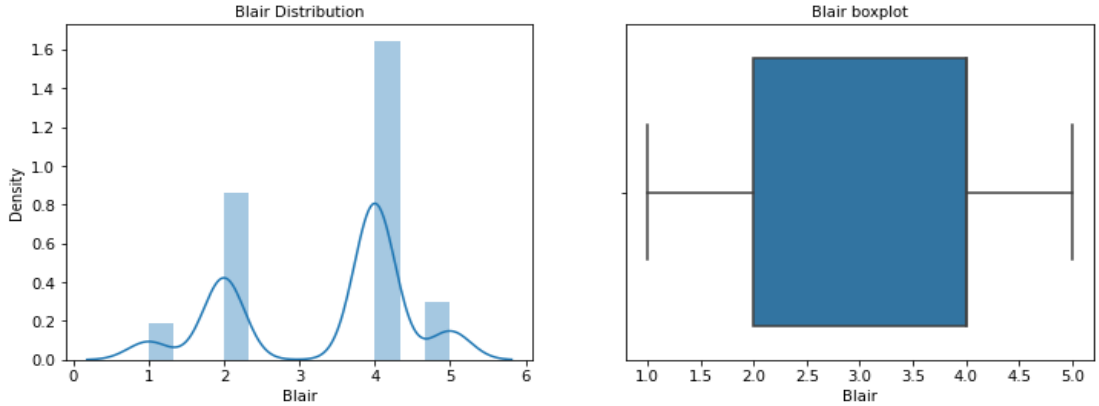
The data points seem to be normally distributed with barely any outliers.

- Economic.cond.household



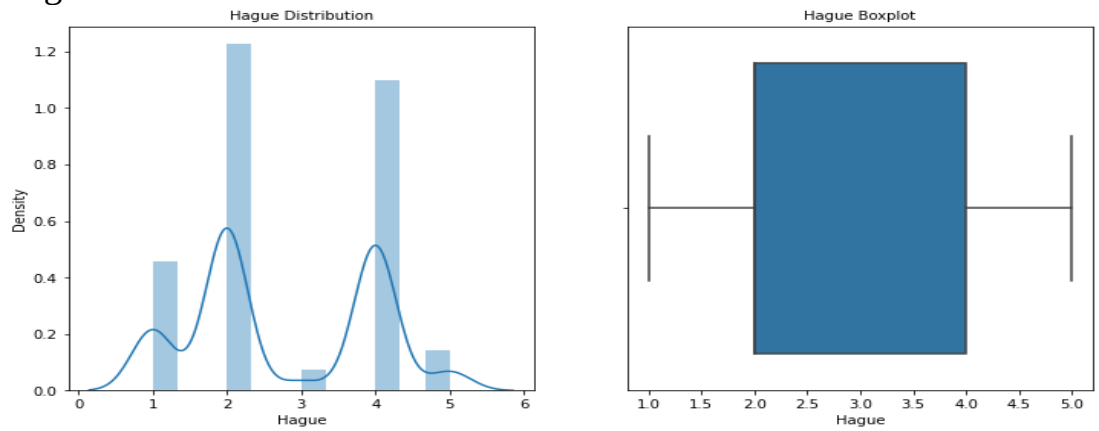
The data points seem to be normally distributed with barely any outliers.

- Blair



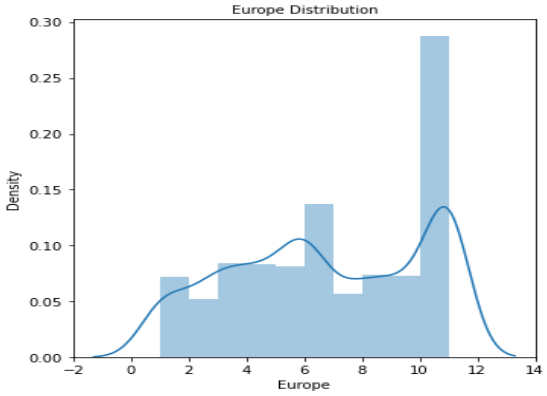
The data points seem to be normally distributed without any outliers.

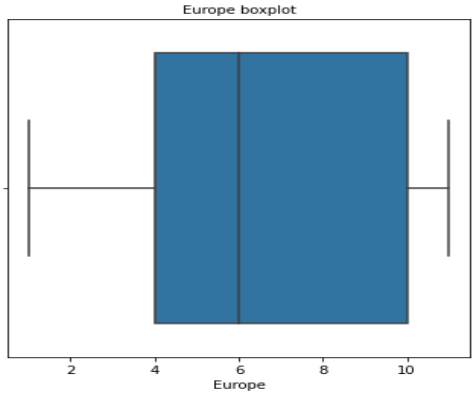
- Hague



The data points seem to be normally distributed without any outliers

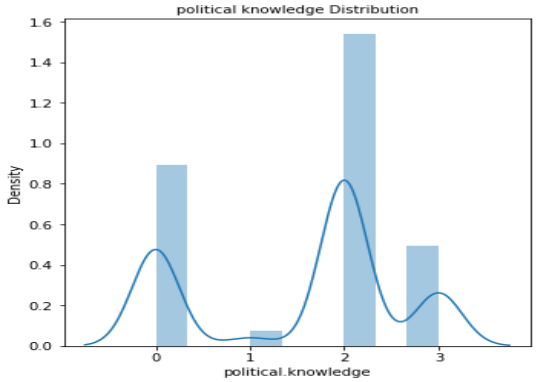
- Europe

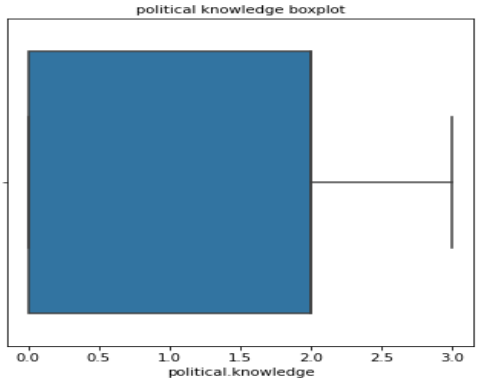




The data points seem to be normally distributed without any outliers.

- Political.knowledge



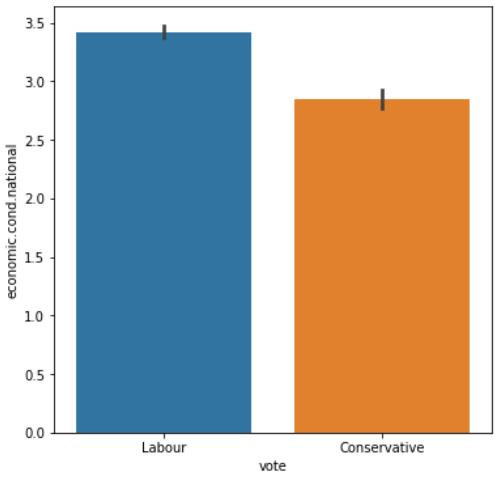


The data points seem to be normally distributed without any outliers.

Bivariate Analysis

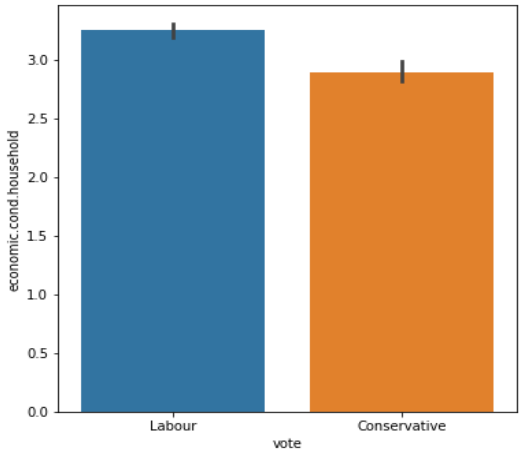
- Vote vs Economic.cond.national

It is seen that people from the Labour party have given higher assessment value of the national economic conditions when compared to those from the Conservative party.



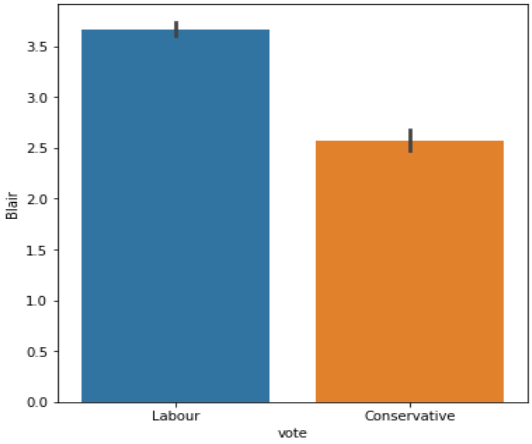
- Vote vs Economic.cond.household

  It is seen that people from the Labour party have given higher assessment value of the national household conditions when compared to those from the Conservative party.



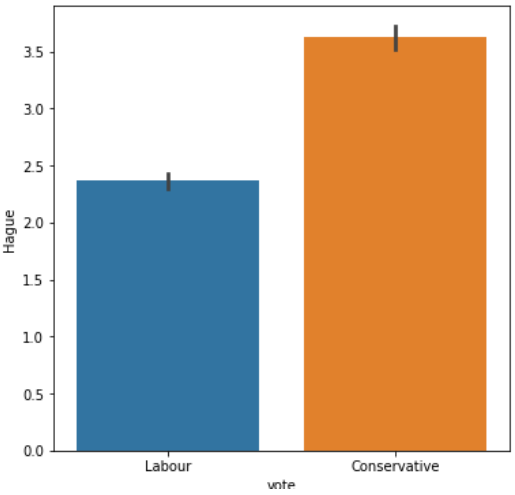
- Vote vs Blair

  It is seen that people from the Labour party have given higher assessment value to the Labour party leader when compared to those from the Conservative party.



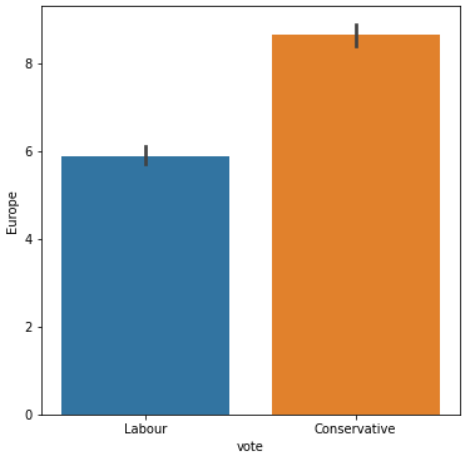
- Vote vs Hague

  It is seen that people from the Conservative party have given higher assessment value to the Conservative party leader when compared to those from the Labour party.



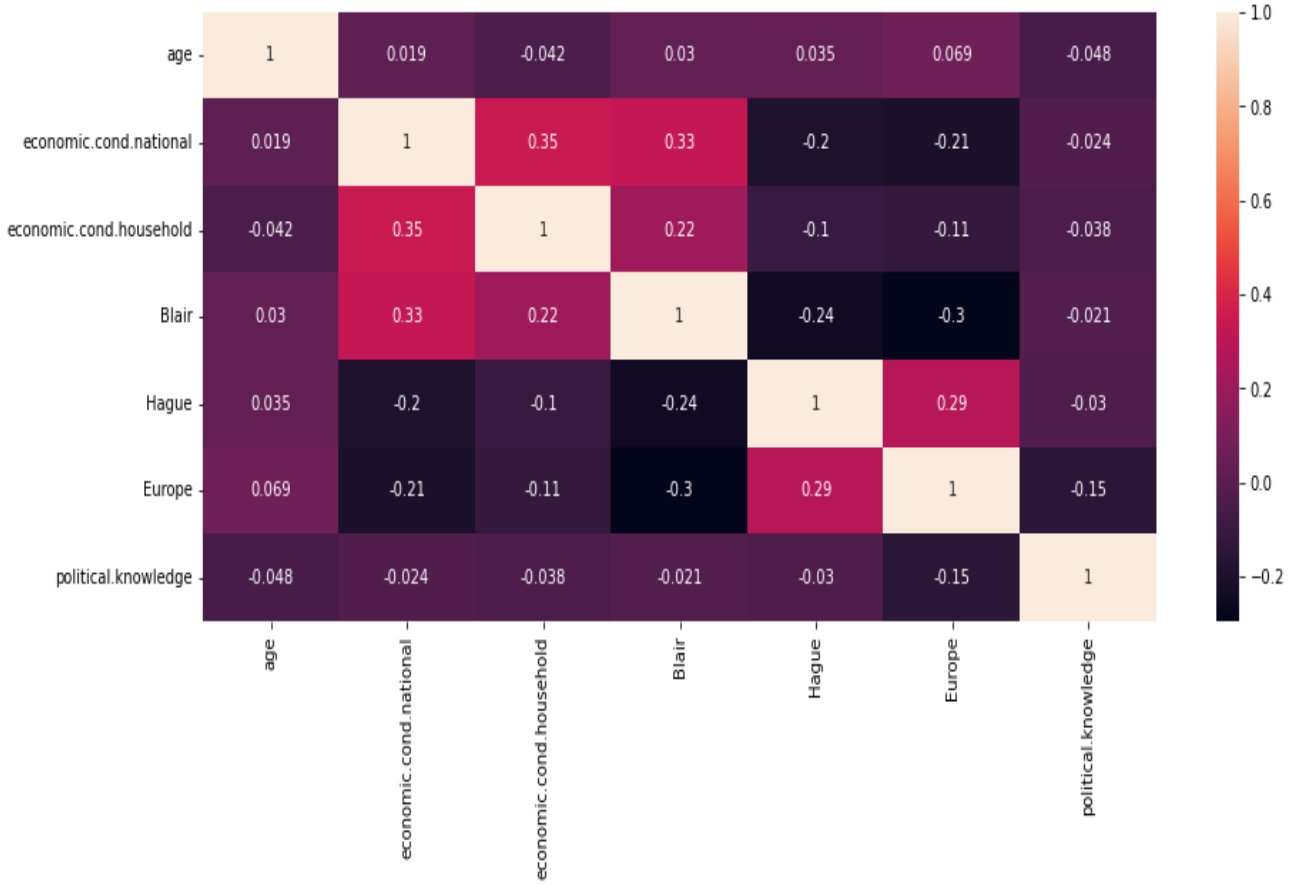
- Vote vs Europe

It is seen that people from the Conservative party have given higher assessment value portraying 'Eurosceptic' sentiment.



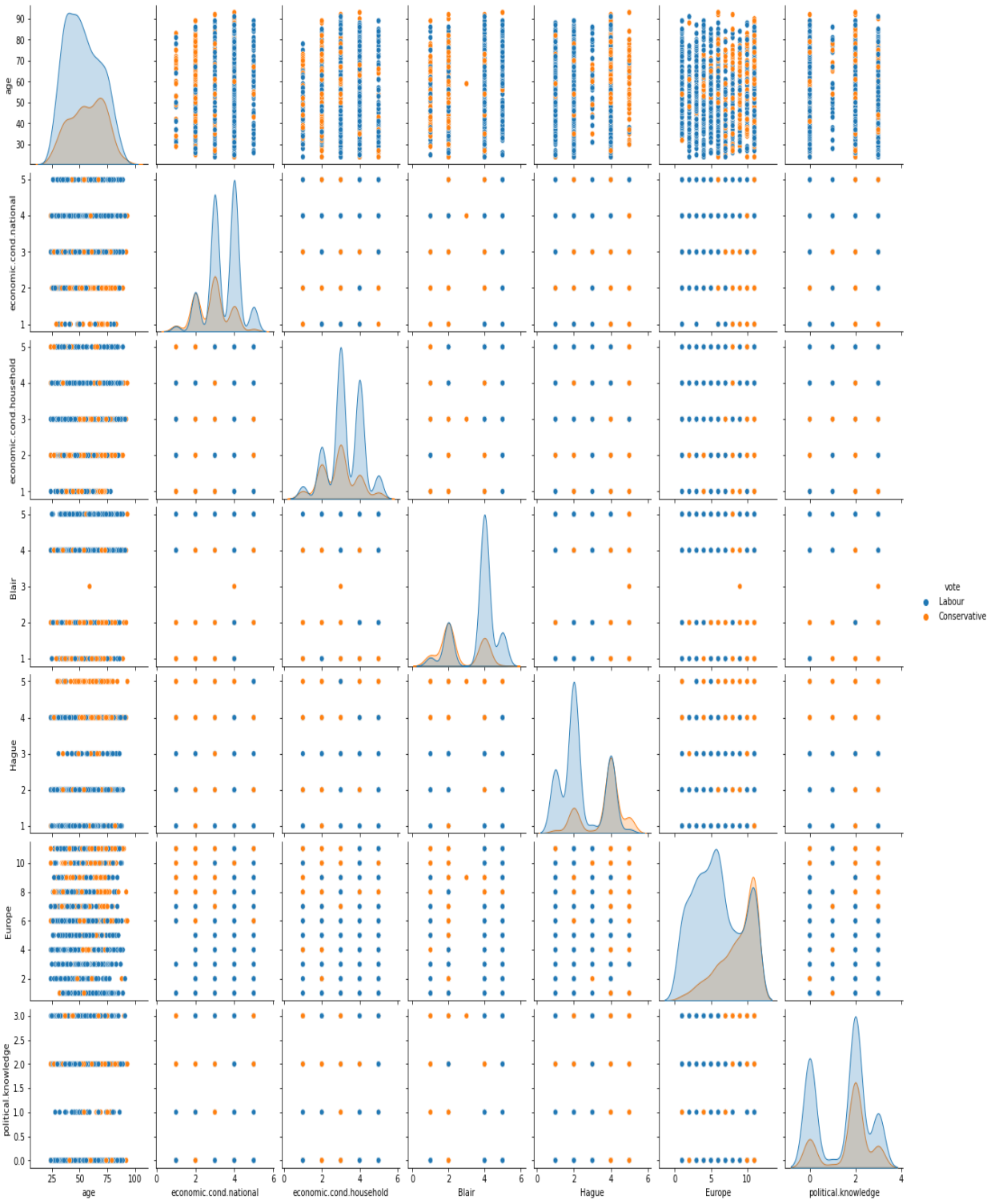
Multivariate Analysis

- Heatmap



There is no strong positive correlation between any of the variables.

- Pairplot



In this plot we can understand the relationship between all the numerical variables in the dataset using ‘vote’ variable as the reference and establish the trends in the datase

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)

The data was encoded in the Jupyter file and its headset is shown below. The vote column was renamed to 'Is_a_Labour_or_not' and the gender column was renamed to 'Is_a_Male_or_not' which suggests that string values 'labour' and 'male' were encoded to numeric value 1.

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	Is_a_Labour_or_not	Is_a_Male_or_not
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

Scaling is not required in the given dataset as the nature of the data points for the independent variables in ordinal in nature and can affect the model's efficiency. However, scaling as adopted only for building of the KNN model which can be seen later in the report.

The below code was written and executed to split the data.

```
X=data_df.drop('Is_a_Labour_or_not',axis=1)
Y=data_df.pop('Is_a_Labour_or_not')
```

```
X_train,X_test,y_train,y_test=train_test_split(X,Y,test_size=.30,random_state=0)
```

.

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

Logistic regression

Basic Logistic regression model

Logistic regression was applied to the model.

```
#LOGISTIC REGRESSION
Logistic_model = LogisticRegression()
Logistic_model.fit(X_train, Y_train)
```

```
LogisticRegression()
```

Classification Report of the training data:

	precision	recall	f1-score	support
0	0.77	0.66	0.71	328
1	0.86	0.91	0.89	739
accuracy			0.84	1067
macro avg	0.82	0.79	0.80	1067
weighted avg	0.83	0.84	0.83	1067

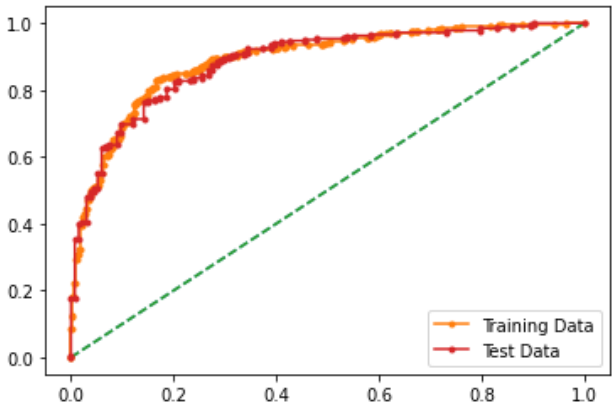
Classification Report of the test data:

	precision	recall	f1-score	support
0	0.72	0.70	0.71	134
1	0.88	0.89	0.88	324
accuracy			0.83	458
macro avg	0.80	0.80	0.80	458
weighted avg	0.83	0.83	0.83	458

confusion matrix

Text(0.5, 1.0, 'Test Data')

AUC for the Training Data: 0.890
AUC for the Test Data: 0.887



Hyperparameter tuned Logistic regression model

```
logreg = LogisticRegression()
clf = GridSearchCV(logreg,
                   param_grid = parameters,
                   scoring='accuracy',
                   cv=10)
clf.fit(X_train,Y_train)
```

```
GridSearchCV(cv=10, estimator=LogisticRegression(),
             param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
                         'penalty': ['l1', 'l2'],
                         'solver': ['newton-cg', 'lbfgs', 'liblinear']},
             scoring='accuracy')
```

Classification Report of the training data:

	precision	recall	f1-score	support
0	0.77	0.66	0.71	328
1	0.86	0.91	0.89	739
accuracy			0.84	1067
macro avg	0.82	0.79	0.80	1067
weighted avg	0.83	0.84	0.83	1067

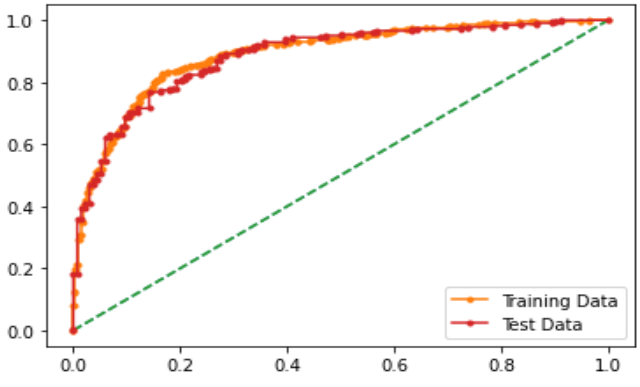
Classification Report of the test data:

	precision	recall	f1-score	support
0	0.73	0.69	0.71	134
1	0.88	0.90	0.89	324
accuracy			0.84	458
macro avg	0.80	0.79	0.80	458
weighted avg	0.83	0.84	0.83	458

confusion matrix

Text(0.5, 1.0, 'Test Data')

AUC for the Training Data: 0.890
AUC for the Test Data: 0.886



Linear discriminant analysis

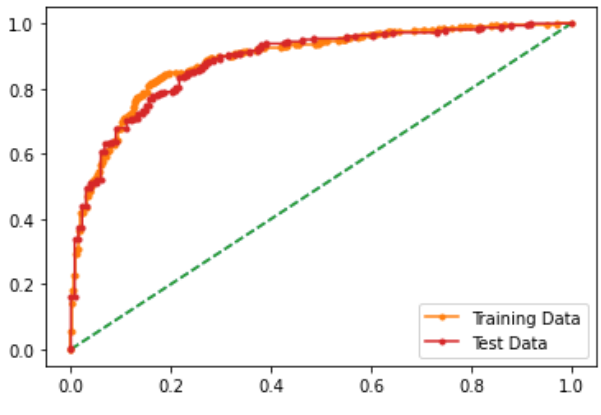
```
#LDA
clf = LinearDiscriminantAnalysis()
ldamodel=clf.fit(X_train, Y_train)
ldamodel
```

```
LinearDiscriminantAnalysis()
```

```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       0.75      0.68      0.71       328
           1       0.86      0.90      0.88       739

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


Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.72      0.72      0.72       134
           1       0.88      0.89      0.88       324

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

confusion matrix


```
AUC and ROC FOR LDA
AUC for the Training Data: 0.890
AUC for the Test Data: 0.886
```



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

Naïve Bayes Model

```
#Naïve Bayes Model

NB_model=GaussianNB()
NB_model.fit(X_train, Y_train)
GaussianNB()
```

```
GaussianNB()
```

Confusion Matrix

Train data

```
[[235  93]
 [ 79 660]]
```

Test data

```
[[ 99  35]
 [ 45 279]]
```

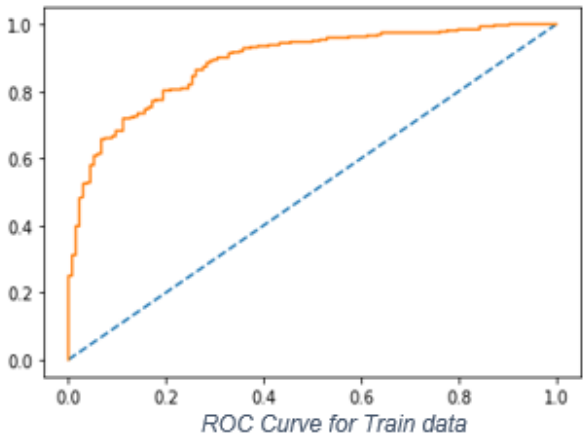
Classification report for train data

	precision	recall	f1-score	support
0	0.75	0.72	0.73	328
1	0.88	0.89	0.88	739
accuracy			0.84	1067
macro avg	0.81	0.80	0.81	1067
weighted avg	0.84	0.84	0.84	1067

Classification report for test data

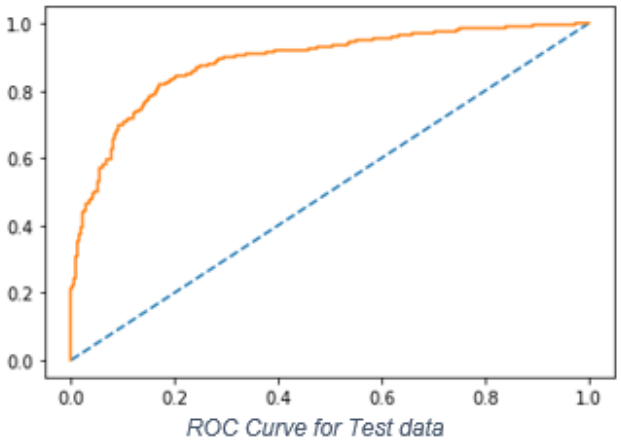
	precision	recall	f1-score	support
0	0.69	0.74	0.71	134
1	0.89	0.86	0.87	324
accuracy			0.83	458
macro avg	0.79	0.80	0.79	458
weighted avg	0.83	0.83	0.83	458

```
the auc  0.887

[<matplotlib.lines.Line2D at 0x1a43e66c5c8>]
```



ROC Curve for Train data

```
the auc 0.887

[<matplotlib.lines.Line2D at 0x1a3d75ecb08>]
```



ROC Curve for Test data

KNN Model

```
#KNN
x=df2.drop("Is_a_Labour_or_not",axis=1)
y=df2.pop('Is_a_Labour_or_not')
```

Scaling was done in case of KNN model so that the loss function is not biased to a particular feature.

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

The data is split after scaling.

```
#KNN
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y, random_state=1)
```

Basic KNN model

```
#KNN

from sklearn.neighbors import KNeighborsClassifier
KNN_model=KNeighborsClassifier()
KNN_model.fit(x_train,y_train)
```

```
KNeighborsClassifier()
```

Confusion Matrix

Train data

```
[[263  88]
 [ 63 729]]
```

Test data

```
[[ 81  30]
 [ 37 234]]
```

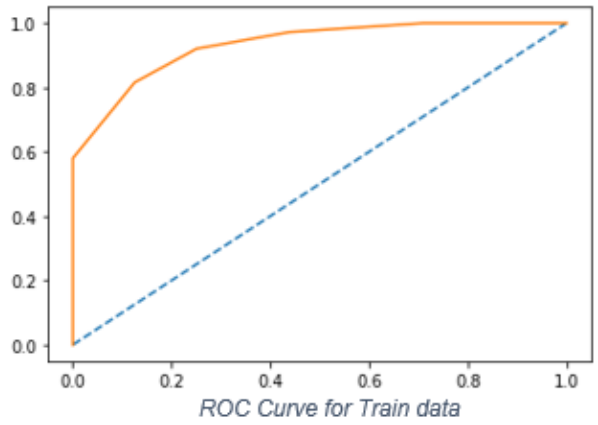
Classification report for train data

	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

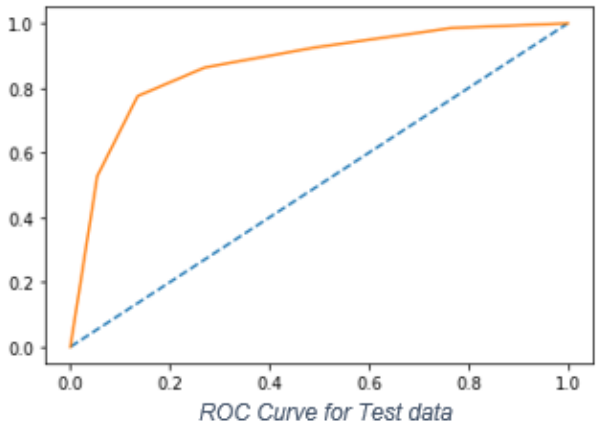
Classification report for test data

	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

```
the auc 0.932
```



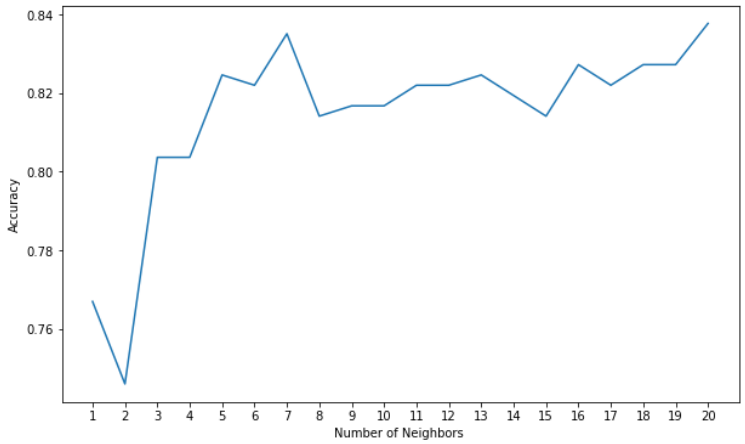

```
the auc curve 0.870

[<matplotlib.lines.Line2D at 0x1a3d9b17248>]
```



The accuracy value for k=1 to k=20 is below:

```
array([0.76701571, 0.7460733 , 0.80366492, 0.80366492, 0.82460733,
       0.82198953, 0.83507853, 0.81413613, 0.81675393, 0.81675393,
       0.82198953, 0.82198953, 0.82460733, 0.81937173, 0.81413613,
       0.82722513, 0.82198953, 0.82722513, 0.82722513, 0.83769634])
```

The graph of accuracy vs number of neighbors is as:



Hyperparameter tuned KNN model

K values from 9 to 14 were used as their accuracies show a stable pattern without major rise or dip in the plot.

```
grid_params = { 'n_neighbors' : [9,10,11,12,13,14],
                'weights' : ['uniform','distance'],
                'metric' : ['minkowski','euclidean','manhattan']}
gs = GridSearchCV(KNeighborsClassifier(), grid_params, verbose = 1, cv=500, n_jobs = -1)
g_res = gs.fit(x_train, y_train)
```

```
Fitting 500 folds for each of 36 candidates, totalling 18000 fits
```

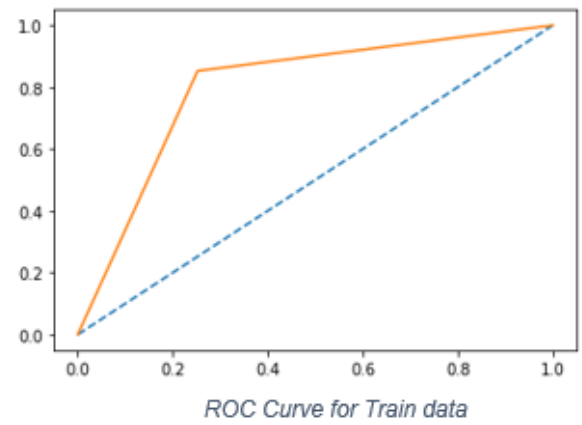
Classification report for train data

	precision	recall	f1-score	support
0	0.75	0.75	0.75	351
1	0.89	0.89	0.89	792
accuracy			0.85	1143
macro avg	0.82	0.82	0.82	1143
weighted avg	0.85	0.85	0.85	1143

Classification report for test data

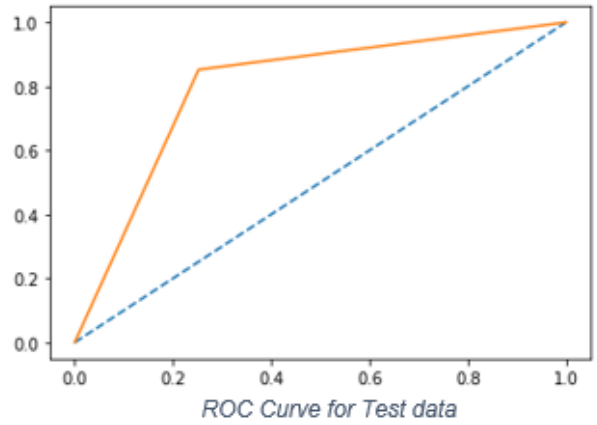
	precision	recall	f1-score	support
0	0.67	0.75	0.71	111
1	0.89	0.85	0.87	271
accuracy			0.82	382
macro avg	0.78	0.80	0.79	382
weighted avg	0.83	0.82	0.82	382

```
Auc  0.909
[<matplotlib.lines.Line2D at 0x1a43e5ebf48>]
```



ROC Curve for Train data

```
Auc  0.891
[<matplotlib.lines.Line2D at 0x1a44740d548>]
```



ROC Curve for Test data

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

Bagging

Basic Bagging Model

```
cart=RandomForestClassifier()
Bagging_model=BaggingClassifier(base_estimator=cart,n_estimators=100, random_state=1)

Bagging_model.fit(X_train,Y_train)
```

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

Classification report for train data

	precision	recall	f1-score	support
0	0.97	0.92	0.94	332
1	0.96	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

Classification report for test data

	precision	recall	f1-score	support
0	0.71	0.71	0.71	130
1	0.88	0.89	0.89	328
accuracy			0.84	458
macro avg	0.80	0.80	0.80	458
weighted avg	0.84	0.84	0.84	458

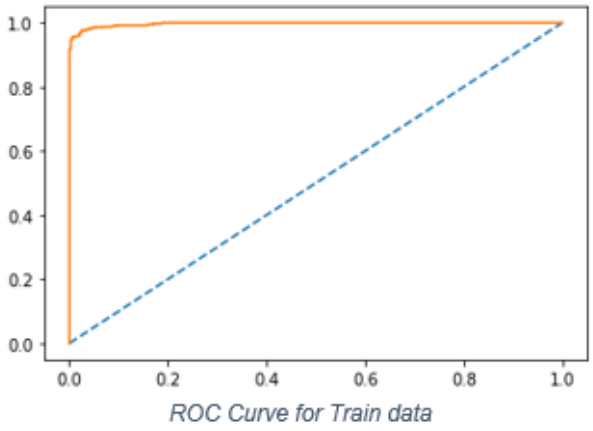
Confusion Matrix

Train data

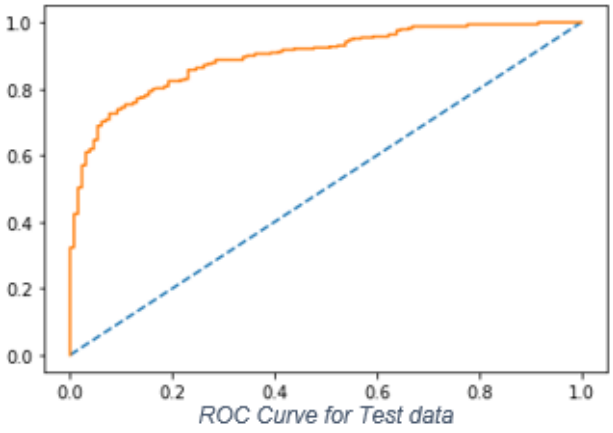
```
[[304  28]
 [  9 726]]
```

Test data

```
[[ 92  38]
 [ 37 291]]
```



ROC Curve for Train data



ROC Curve for Test data

Hyperparameter tuned Bagging Model

```
param_grid = {'base_estimator__max_depth' : [1,2,3,4,5],'max_samples' : [0.05, 0.1, 0.2, 0.5]}

clf = GridSearchCV(BaggingClassifier(RandomForestClassifier(),n_estimators=400, random_state=1),param_grid)
clf.fit(X_train, Y_train)
```

```
GridSearchCV(estimator=BaggingClassifier(base_estimator=RandomForestClassifier(),
                                         n_estimators=400, random_state=1),
             param_grid={'base_estimator__max_depth': [1, 2, 3, 4, 5],
                         'max_samples': [0.05, 0.1, 0.2, 0.5]})
```

Classification report for train data

	precision	recall	f1-score	support
0	0.82	0.61	0.70	328
1	0.85	0.94	0.89	739
accuracy			0.84	1067
macro avg	0.83	0.78	0.80	1067
weighted avg	0.84	0.84	0.83	1067

Classification report for test data

	precision	recall	f1-score	support
0	0.80	0.67	0.73	134
1	0.87	0.93	0.90	324
accuracy			0.86	458
macro avg	0.84	0.80	0.82	458
weighted avg	0.85	0.86	0.85	458

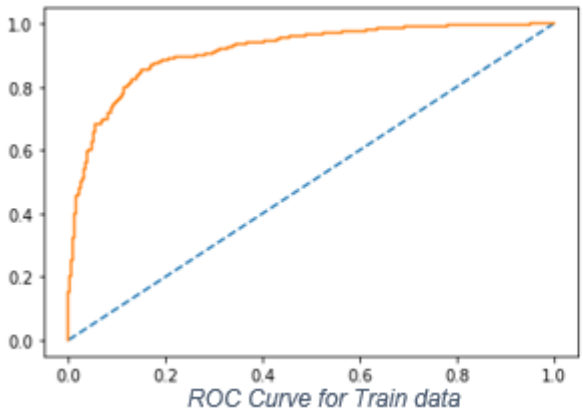
Confusion Matrix

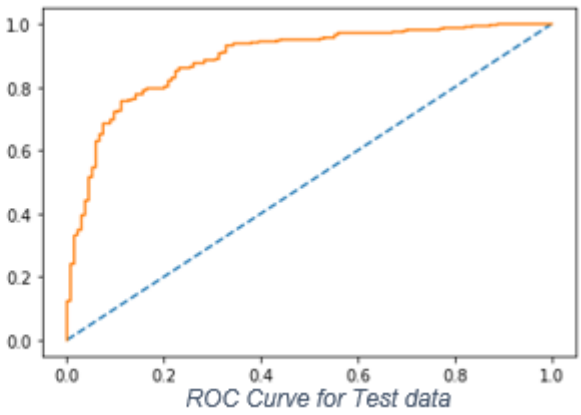
Train data

```
[[201 127]
 [ 44 695]]
```

Test data

```
[[ 90  44]
 [ 22 302]]
```





Boosting

ADA Boosting model

Classification report for train data

	precision	recall	f1-score	support
0	0.78	0.72	0.74	332
1	0.88	0.91	0.89	735
accuracy			0.85	1067
macro avg	0.83	0.81	0.82	1067
weighted avg	0.84	0.85	0.85	1067

Classification report for test data

	precision	recall	f1-score	support
0	0.28	0.31	0.29	130
1	0.71	0.68	0.70	328
accuracy			0.58	458
macro avg	0.50	0.50	0.49	458
weighted avg	0.59	0.58	0.58	458

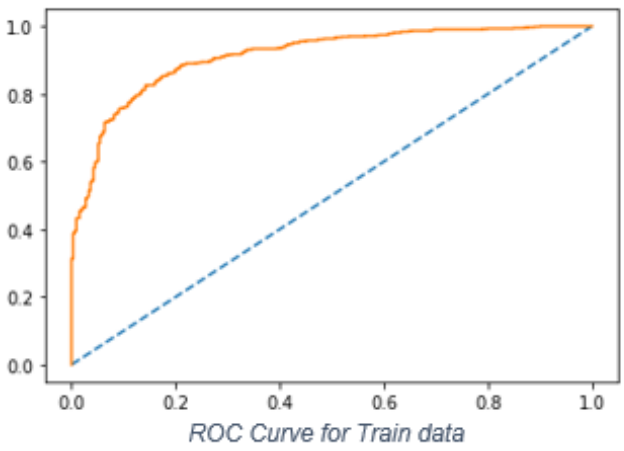
Confusion Matrix

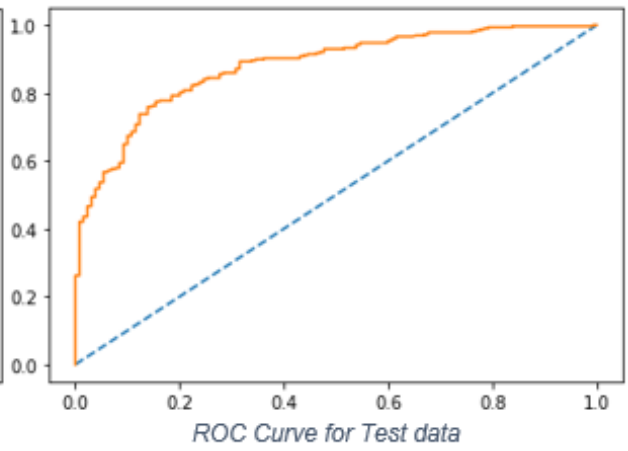
Train data

```
[[238  94]
 [ 69 666]]
```

Test data

```
[[ 40  90]
 [104 224]]
```





Gradient Boosting model

Classification report for train data

	precision	recall	f1-score	support
0	0.84	0.79	0.81	332
1	0.91	0.93	0.92	735
accuracy			0.89	1067
macro avg	0.87	0.86	0.87	1067
weighted avg	0.89	0.89	0.89	1067

Classification report for test data

	precision	recall	f1-score	support
0	0.28	0.31	0.29	130
1	0.71	0.68	0.70	328
accuracy			0.58	458
macro avg	0.50	0.50	0.49	458
weighted avg	0.59	0.58	0.58	458

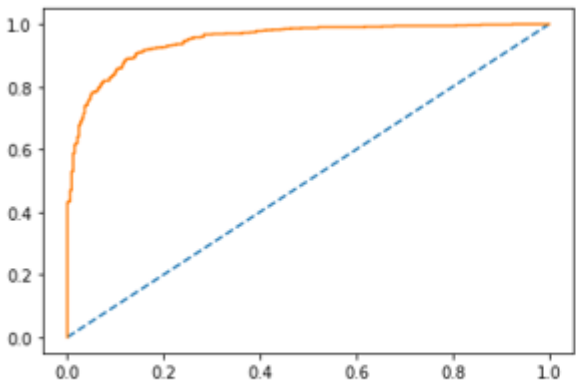
Confusion Matrix

Train data

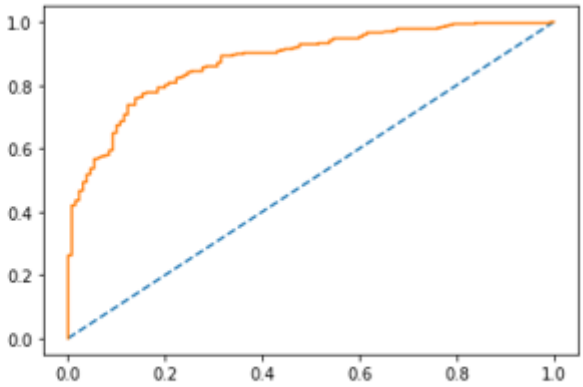
```
[[ 95 237]
 [219 516]]
```

Test data

```
[[ 99  35]
 [ 45 279]]
```



ROC Curve for Train data



ROC Curve for Test data

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

In the above table, the training and test values for Logistic regression model, KNN model ad Bagging model are taken from their hyper parameter tuned model.

	Training Data			Test Data		
	Recall value	Accuracy	AUC	Recall value	Accuracy	AUC
Logistic Regression Model	0.91	0.84	0.89	0.9	0.84	0.886
LDA Model	0.9	0.83	0.89	0.89	0.84	0.886
Naïve Bayes Model	0.89	0.84	0.887	0.86	0.83	0.825
KNN Model	0.89	0.85	0.909	0.85	0.82	0.891
Bagging Model	0.94	0.93	0.914	0.83	0.86	0.891
ADA Boosting Model	0.91	0.85	0.913	0.68	0.58	0.879
Gradient Boosting Model	0.93	0.89	0.939	0.86	0.83	0.931

It can be seen that the recall values for both train and test data are high in case of- Logistic regression model, LDA model, Naïve Bayes model, Bagging model and Gradient Boosting model. These are good fit models.

In case of ADA model, even though the recall train value is high, it's not an optimized model due its low recall test value (0.68).

1.8 Based on these predictions, what are the insights.

- From the heatmap it is seen that there are no strong correlations so the values need to be more practical for analysis.
- It is seen that the labour party has given higher assessment value to national economic conditions as compared to conservative party. This means that the conservative party is not satisfied with the current states and this needs to be looked into. Maybe a detailed survey can be sent to this party.
- It is seen that the labour party has given higher assessment value to national household conditions as compared to conservative party. This can be dealt with by providing them with the kind of household they would prefer.

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.

Document: '1941-Roosevelt.txt'
The number of characters is 7571
The number of sentences is 68
The number of words is 1526

Document: '1961-Kennedy.txt'
The number of characters is 7618
The number of sentences is 52
The number of words is 1543

Document: '1973-Nixon.txt'
The number of characters is 9991
The number of sentences is 68
The number of words is 2006

2.2 Remove all the stop words from all three speeches.

Number of words in the Document: '1941-Roosevelt.txt' after removing stop words is: 631
Number of words in the Document: '1961-Kennedy.txt' after removing stop words is: 694
Number of words in the Document: '1973-Nixon.txt' after removing stop words is: 847

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

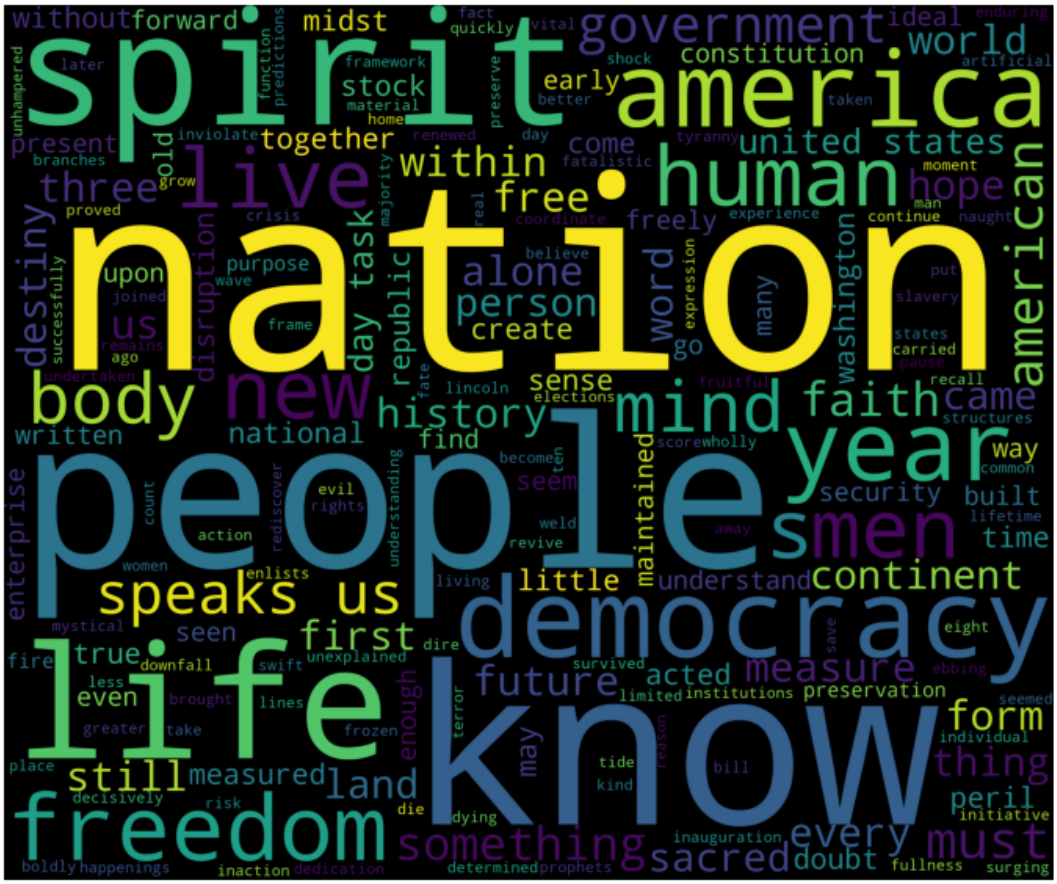
Document: '1941-Roosevelt.txt'
[('nation', 12),
('know', 10),
('spirit', 9)]

Document: ‘1961-Kennedy.txt’
[('let', 16),
('us', 12),
('world', 8)]

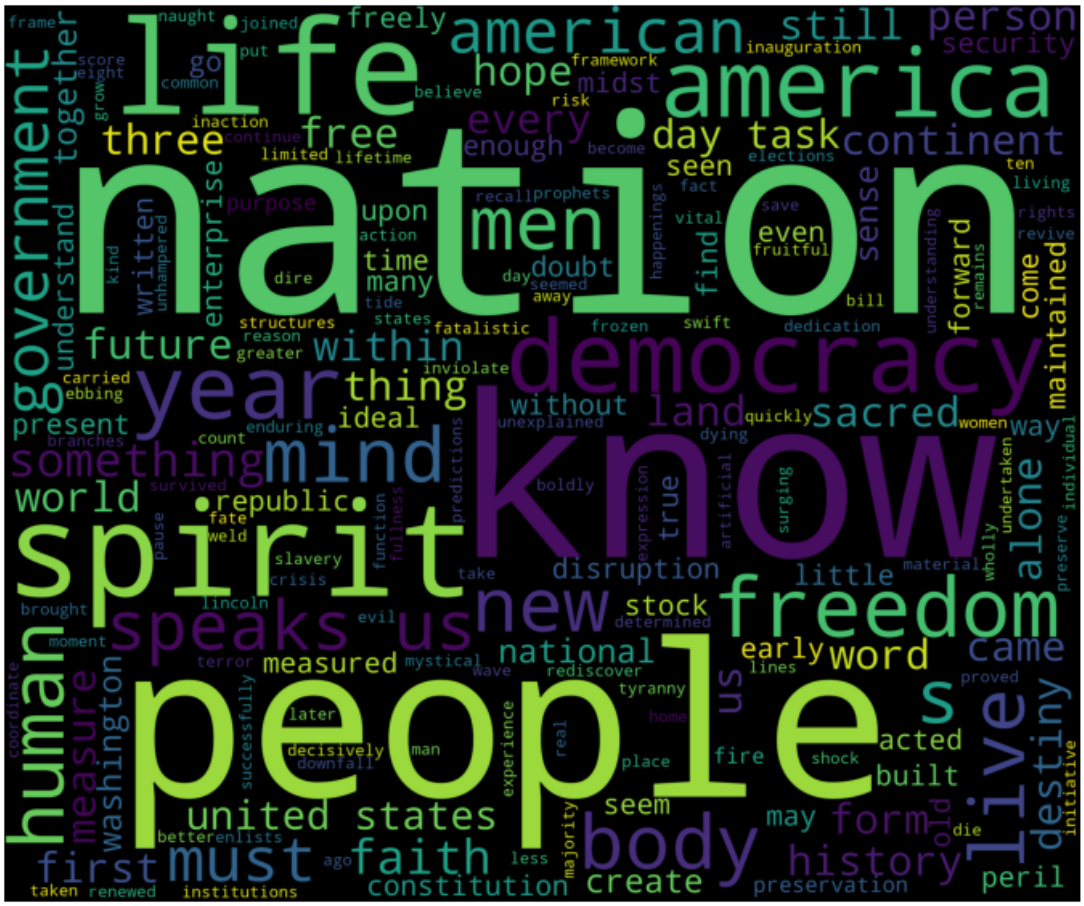
Document: ‘1973-Nixon.txt’
[('us', 26),
('let', 22),
('america', 21)]

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

Document: ‘1941-Roosevelt.txt’



Document: ‘1961-Kennedy.txt’



Document: ‘1973-Nixon.txt’