
Date: 16 January 2022

Great Learning

Authored by: ANIMESH HALDER

Great Learning
POWER AHEAD

# Content

## List of Tables:

## List of Figures:

# Problem 1: Exit Poll

One of the leading news channel CNBE wants to analyze recent elections. This survey was conducted on 1525 voters with 9 variables. The objective is to build a model, to predict which party a voter will vote for on the basis of the given information, and 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 given dataset is in excel format. The dataset 'Election_Data.xlsx' is loaded using pandas 'read_excel' function. The attributes of the dataset has been checked by the pandas head function as given in Figure 1.1.

| | 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 |

Figure 1.1: The first five rows of the dataset, showing the dependent and independent variables.

The shape and the detailed information about the column and row numbers, the data types, memory used, are obtained by using the 'info' function, and the variables in the dataset are of two datatypes.

### Statistical description of the dataset:

The statistical summary for numerical columns are given in the following Figure 1.2

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **age** | 1525.0 | 54.182295 | 15.711209 | 24.0 | 41.0 | 53.0 | 67.0 | 93.0 |
| **economic.cond.national** | 1525.0 | 3.245902 | 0.880969 | 1.0 | 3.0 | 3.0 | 4.0 | 5.0 |
| **economic.cond.household** | 1525.0 | 3.140328 | 0.929951 | 1.0 | 3.0 | 3.0 | 4.0 | 5.0 |
| **Blair** | 1525.0 | 3.334426 | 1.174824 | 1.0 | 2.0 | 4.0 | 4.0 | 5.0 |
| **Hague** | 1525.0 | 2.746885 | 1.230703 | 1.0 | 2.0 | 2.0 | 4.0 | 5.0 |
| **Europe** | 1525.0 | 6.728525 | 3.297538 | 1.0 | 4.0 | 6.0 | 10.0 | 11.0 |
| **political.knowledge** | 1525.0 | 1.542295 | 1.083315 | 0.0 | 0.0 | 2.0 | 2.0 | 3.0 |

Figure 1.2: The statistical description the dataset, showing the range, standard deviation, 25, 50 and 75 percentile values of each continuous variables.

Insights:

1. The dataset appears to be flawless.
2. There are two categorical attributes have been found namely: 'vote', and 'gender'.

3. Seven continuous attributes namely 'age', 'economic.cond.national', 'economic.cond.household', 'Blair', 'Hague', 'Europe', and 'political.knowledge' have been also found.
4. The dataset consists of 1525 rows and 10 columns with one column named as 'Unnamed: 0' found no impact to the dataset.
5. 'Vote' is the target variable.
6. As most values range from 0 to 11, Binning has been performed in subsequent steps to place 'age' in a similar range.

### Null value checking:

Based on Panda's null value check, no null values were found in the dataset as shown in Figure 1.3.

```
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
```

Figure 1.3: No null and missing values have been found in all the independent and dependent variables of the dataset.

### Duplicate value checking:

The Pandas duplicate function returned 8 duplicate values which have been removed.

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

The dataset has been checked for missing or null values and has been found no missing or null values in the dataset.

### Data types, shape of the dataset:

The dataset consists of 1525 rows and 10 columns. The datatypes are illustrated in Figure 1.4.

```
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
```

Figure 1.4: The info of the dataset.

The dataset contains 2 categorical columns ‘vote’ and ‘gender’, all other columns are of integer type. In all columns, 'Unnamed: 0' has been found to be of no importance, thus being removed.

The data have the following number of unique variations as describe in Table 1.1.

Table 1.1: The counts of the data and their variations present in the dataset.

| Name: vote, dtype: int64 | | Name: gender, dtype: int64 | |
|---|---|---|---|
| Conservative | 462 | male | 713 |
| Labour | 1063 | female | 812 |

Exploratory Data Analysis (Univariate / Bivariate) helps to understand the distribution of data in the dataset. The distribution of the categorical variables are evaluated by count plot while continuous variables are evaluated by both distplot and box plot. The distplot represents data distribution of a variable against the density distribution, whereas the boxplot is a standardized way of displaying the distribution of data based on a five number summary and reveals the presence of outliers.

The count plot for vote shows Labour party have Got 1057 votes & Conservative party have got 460 votes (Figure 1.5) while Figure 1.6 shows in the survey 808 female and 709 male have been participated.

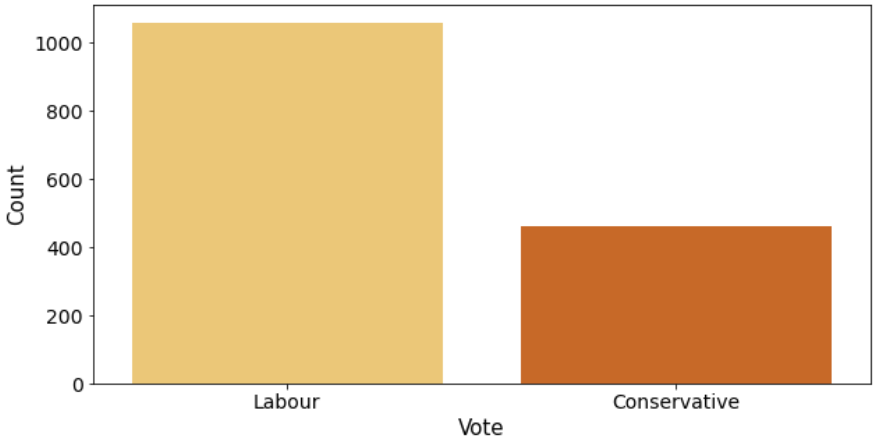


Figure 1.5: The frequency of getting vote in the upcoming election.

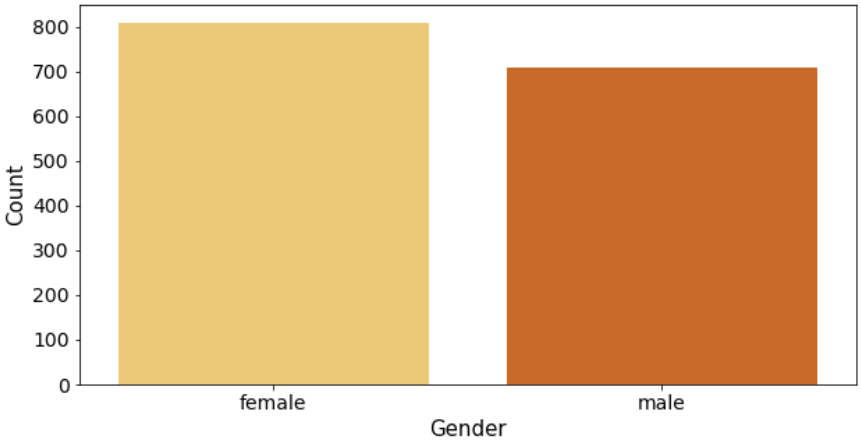


Figure 1.6: The number of female and male participants in the survey.

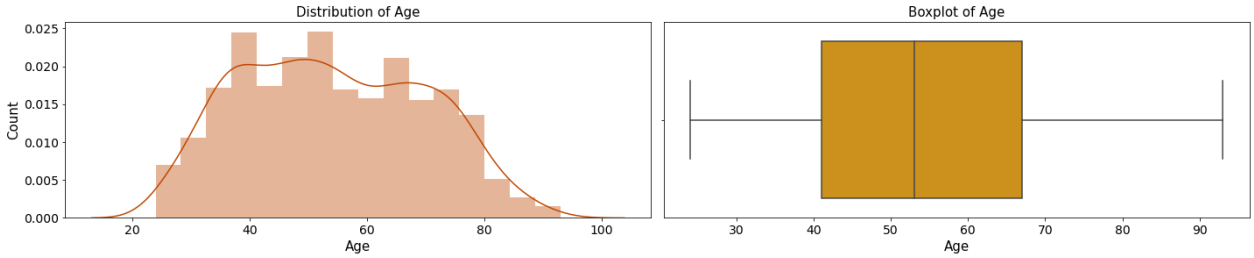


Figure 1.7a: The distplot and boxplot of the variable 'age'.

- The data are normally distributed (skew: 0.139800) from 20 to 90 and have no outliers.

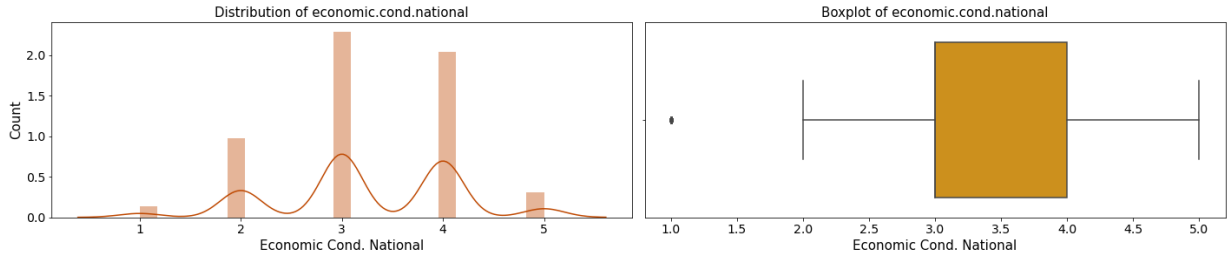


Figure 1.7b: The distplot and boxplot of the variable 'economic cond. national'.

- The distribution is negatively skewed (skew: -0.238474) and the data consists of one outlier.
- Assessment of national economic conditions, 1 to 5.
- The outlier has been found 1, and it is correct hence from a business prospective it will not be a wise decision to treat this outlier.

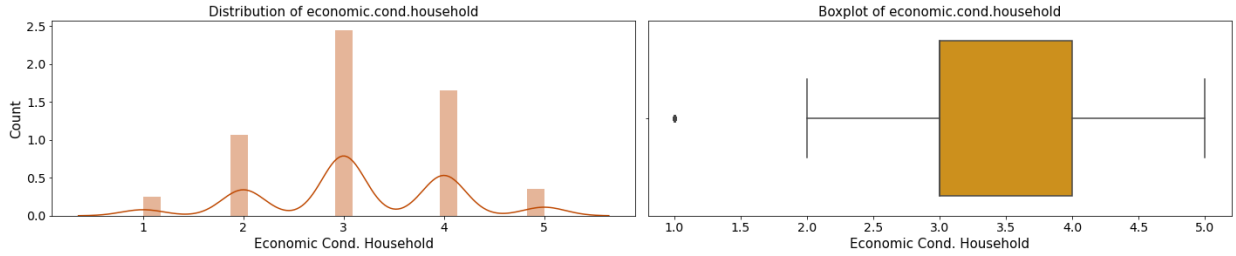


Figure 1.7c: The distplot and boxplot of the variable 'economic cond. household'.

- The distribution is negatively skewed (skew: -0.144148) and the data consists of one outlier.
- Assessment of current household economic conditions, 1 to 5.
- The same argument holds here also, hence this outlier also not be treated.

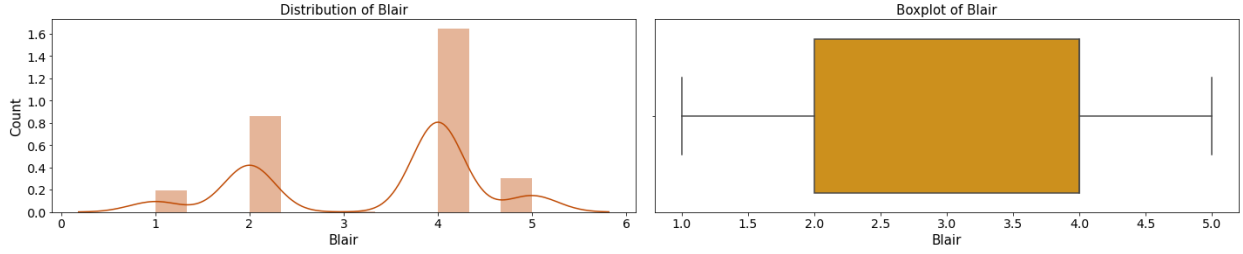


Figure 1.7d: The distplot and boxplot of the variable 'Blair'.

- The distribution is negatively skewed (skew: -0.539514) and the data consists of no outlier.
- Assessment of the Labour leader, 1 to 5.

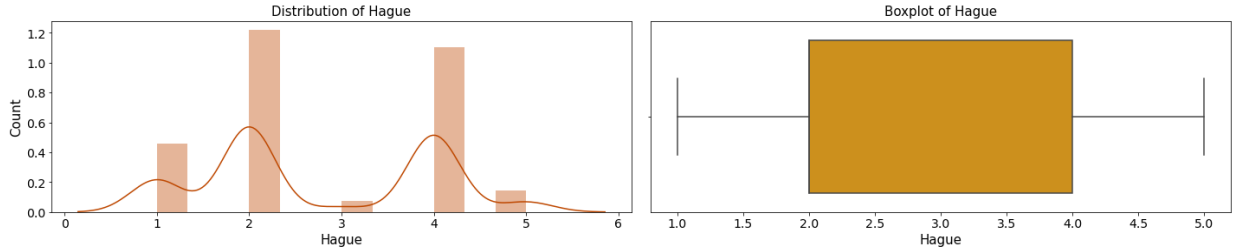


Figure 1.7e: The distplot and boxplot of the variable 'Hague'.

- The distribution is positively skewed (skew: 0.146191) and the data consists of no outlier.
- Assessment of the Conservative leader, 1 to 5.

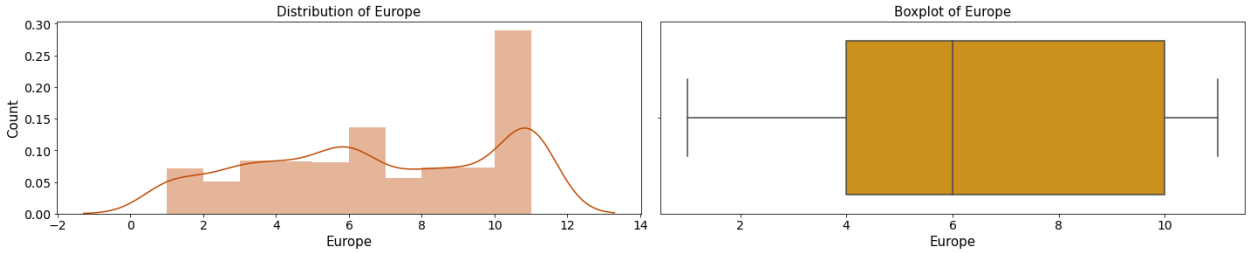


Figure 1.7f: The distplot and boxplot of the variable 'Europe'.

- The distribution is negatively skewed (skew: -0.141891) and the data consists of no outlier.
- This is an 11-point scale that measures respondents' attitudes toward European integration. High scores represent 'Eurosceptic' sentiment.

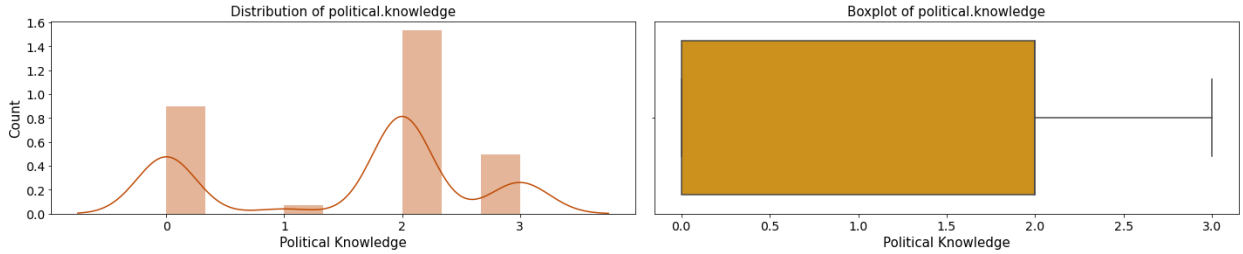


Figure 1.7g: The distplot and boxplot of the variable 'political.knowledge'.

- The distribution is negatively skewed (skew: -0.422928) and the data consists of no outlier.
- Knowledge of parties' positions on European integration, 0 to 3.

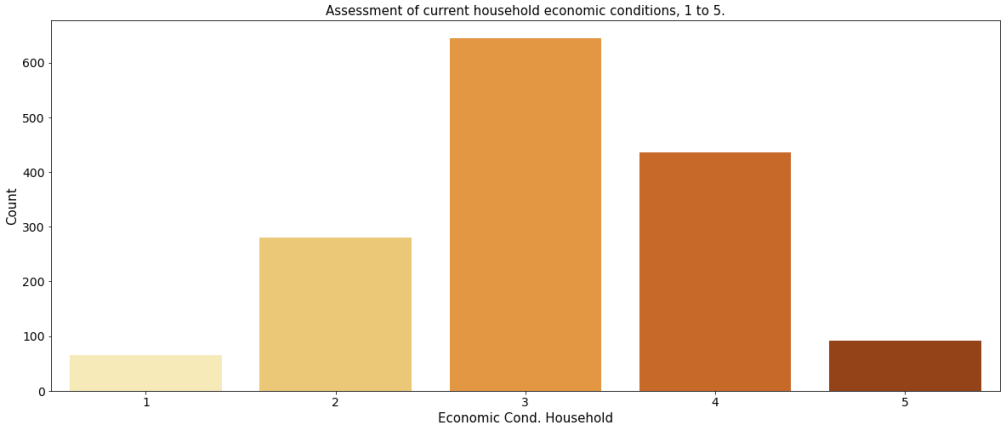


Figure 1.8: The frequency of assessment on current national economy.

- Most frequent rating is 3 and least frequent condition is 1. The average rating is lying between 2 & 3.

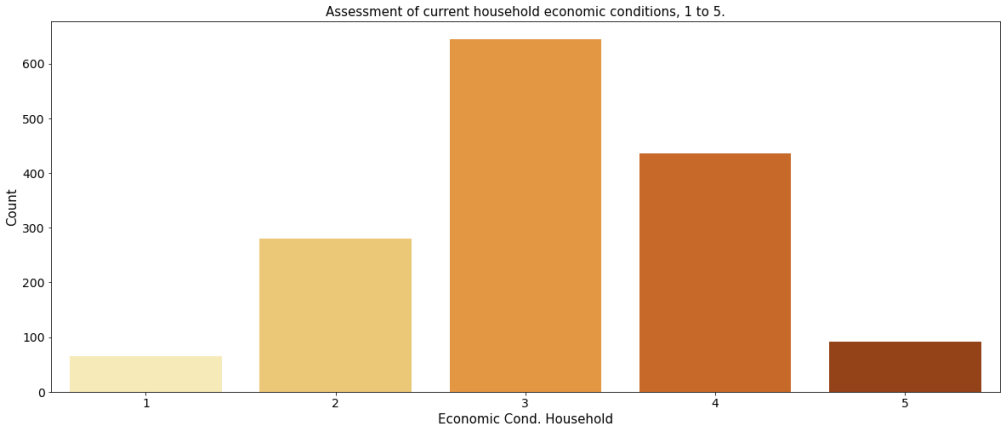


Figure 1.9: The frequency of assessment on current household economy.

- Most frequent rating is 3 and least frequent condition is 1. The average rating is lying between 2 & 3.

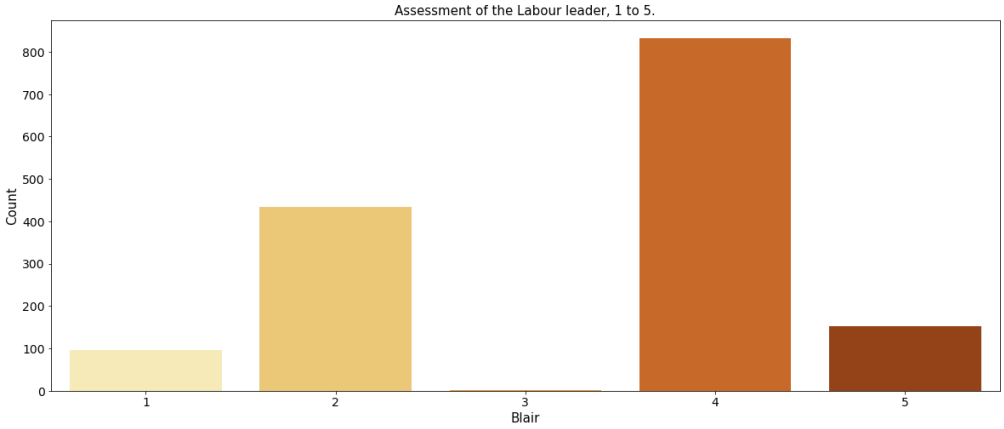


Figure 1.10: The frequency of rating of Blair.

- Labour leader has got most frequent rating of 4 in the surveys.

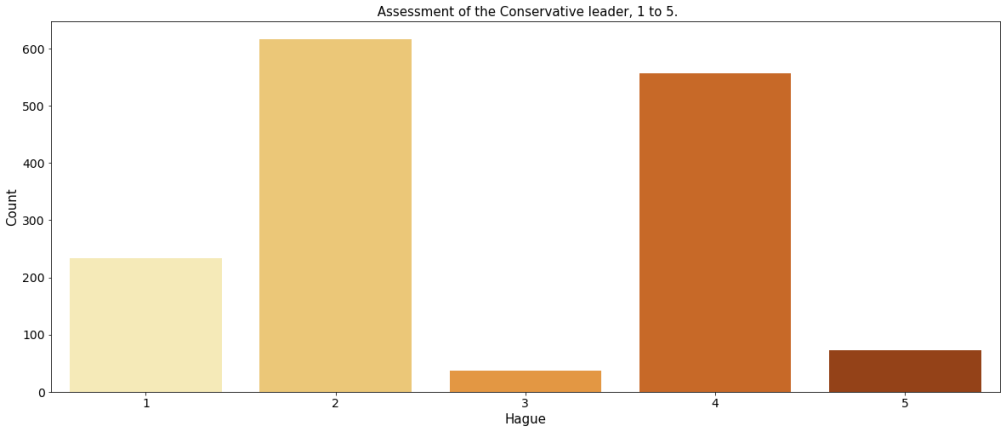


Figure 1.11: The frequency of rating of Hague.

- Conservative leader have got most frequent rating of 2 in the survey.

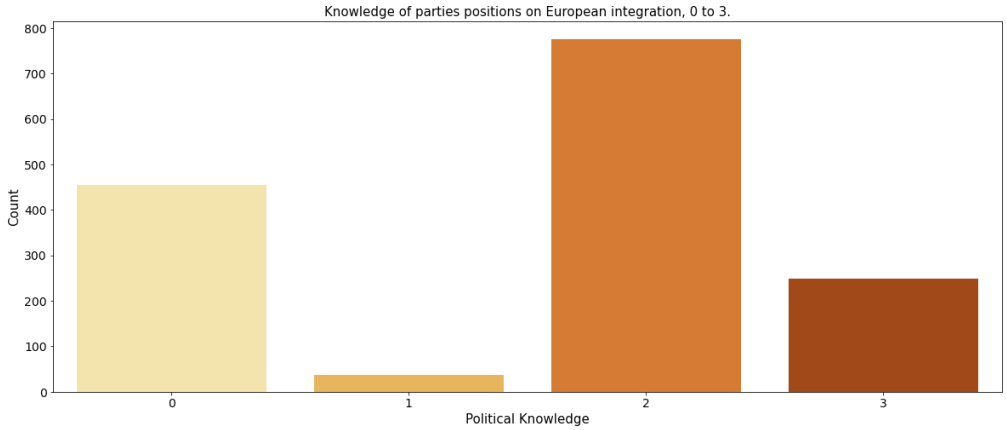


Figure 1.12: The frequency of political knowledge.

- Most frequent scale for political knowledge is 2.

The only variable 'age' has been found in the dataset out of scale in comparison to other variables. Thus binning has been applied to convert discrete age values to ordinal values.

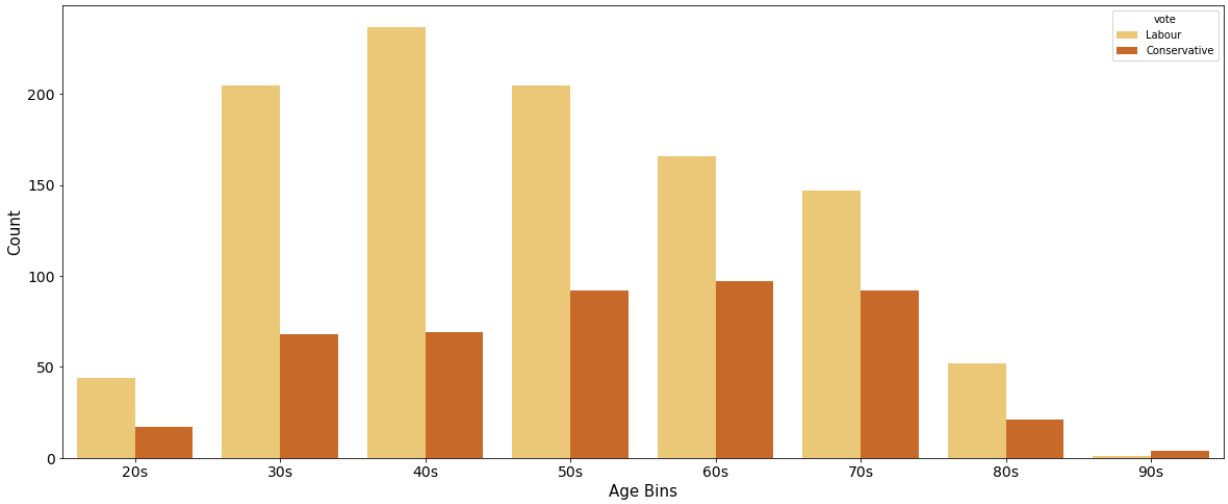


Figure 1.13: Binning of age.

- The age of most of the respondents are in between their 40s & 50s, while 90s are very few.

Bivariate analysis is appropriate to examine the distribution of the continuous variables to categorical variables.

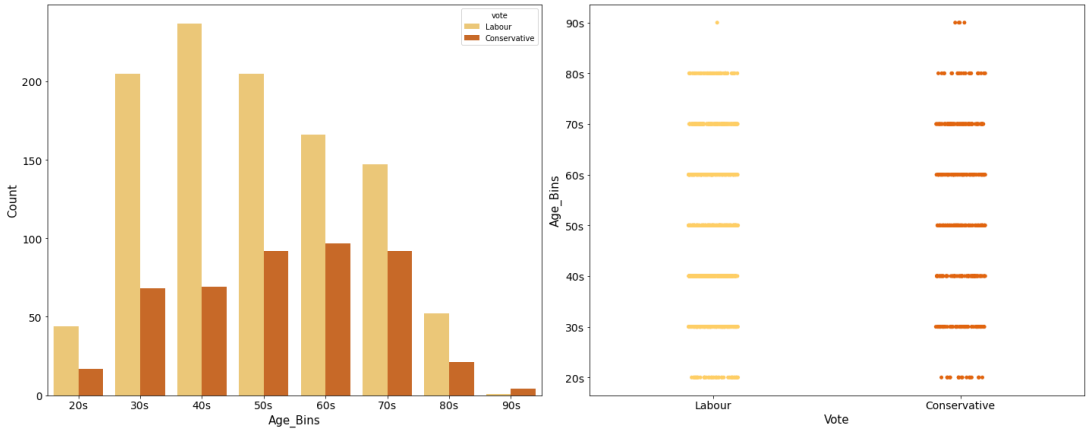


Figure 1.14: Distribution of age to vote.

- Labour is popular with most age groups except for those over 90.

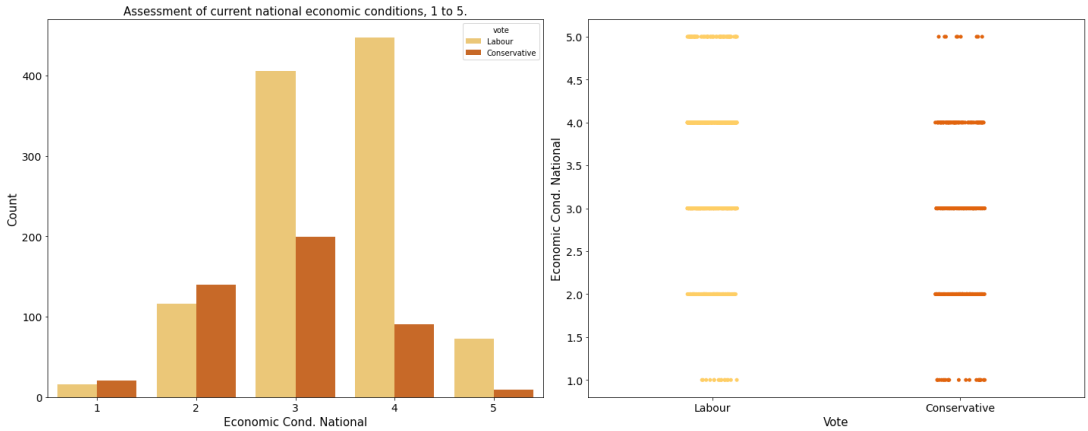


Figure 1.15: Distribution of national economic condition to vote.

- As per above graph Labour party have got most of the great ratings.
- Conservative party fails to score sufficient 5 ratings.

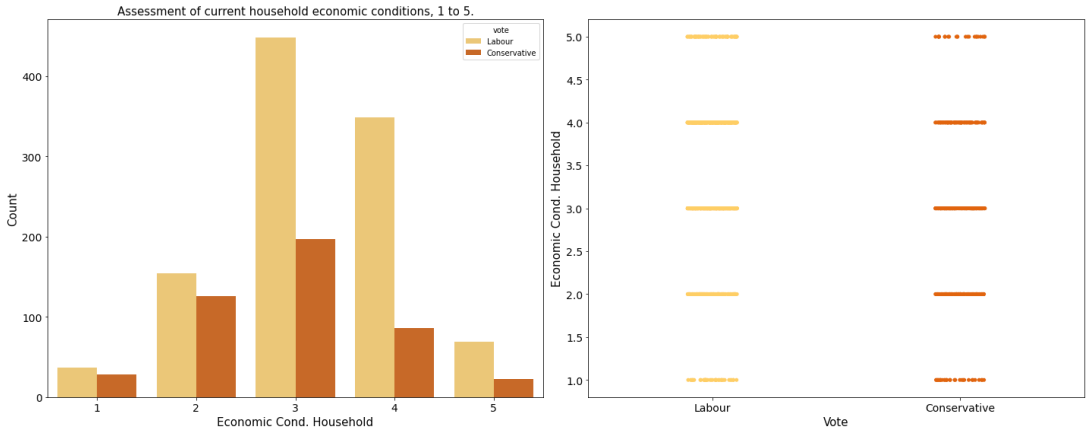


Figure 1.16: Distribution of national economic condition to vote.

- As per above graph Conservative party have got less ratings in comparison to Labour party.
- Conservative party fails to score sufficient 5 ratings but improves in getting 1 score.

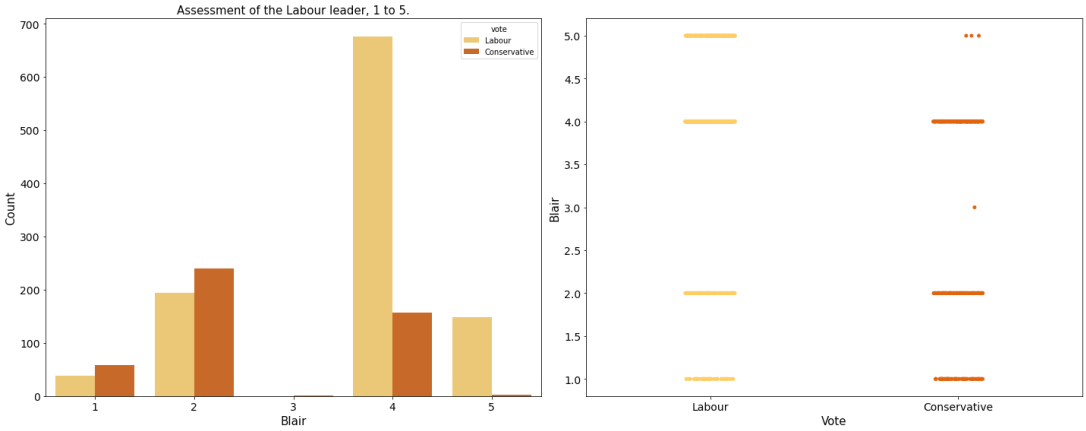


Figure 1.17: Assessment of Labour leader to vote.

- Labour leader have got maximum 4 ratings in their assessment.
- Amongst Labour party leaders which are given average ratings of 3, Conservative party leaders have win there.

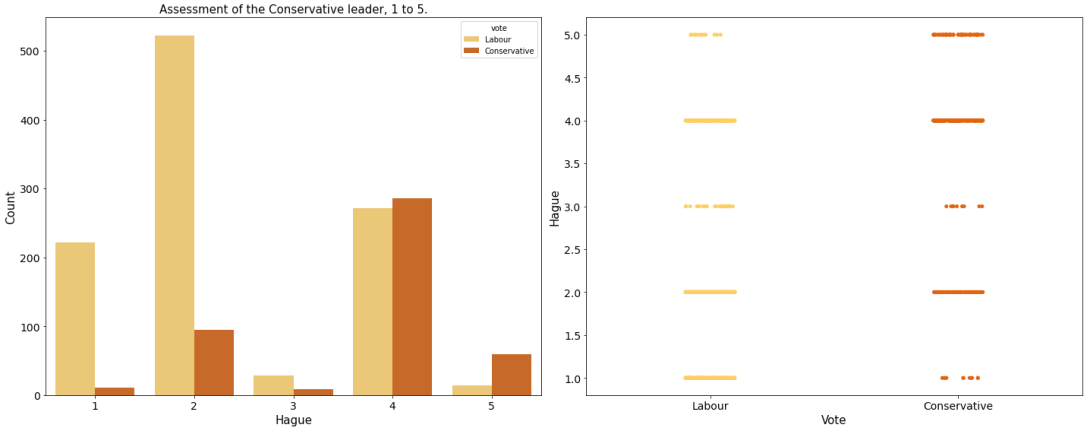


Figure 1.18: Assessment of Conservative leader to vote.

- Conservative leader which are rarely given good ratings as per the graph and vote will be given to Labour party leader.
- Amongst Labour party leaders which are given average ratings of 5, Conservative party leaders have win there.

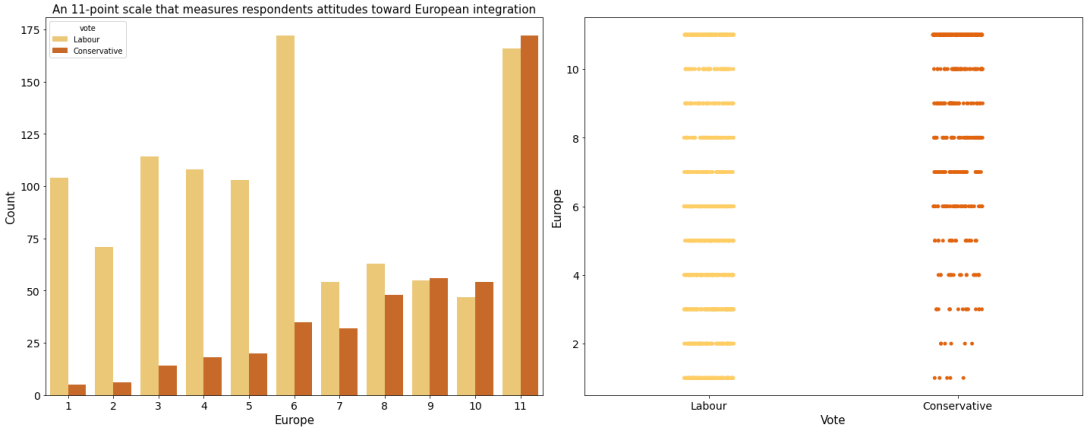


Figure 1.19: Assessment of Eurosceptic attitude to vote.

- Above graph shows that most of the voters have Eurosceptic attitude towards European integrations of Conservative Party.

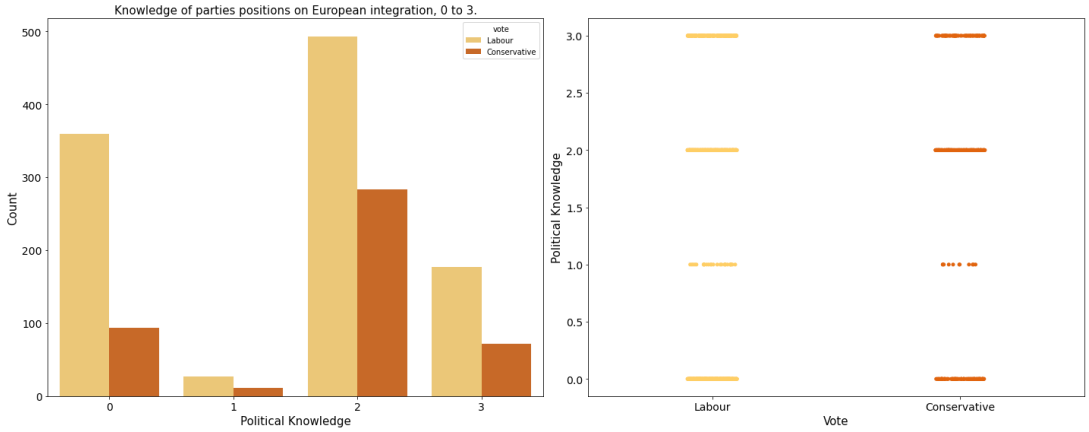


Figure 1.20: Assessment of political knowledge to vote.

- Labour party is having more Europeans integration so they have got most of the votes.

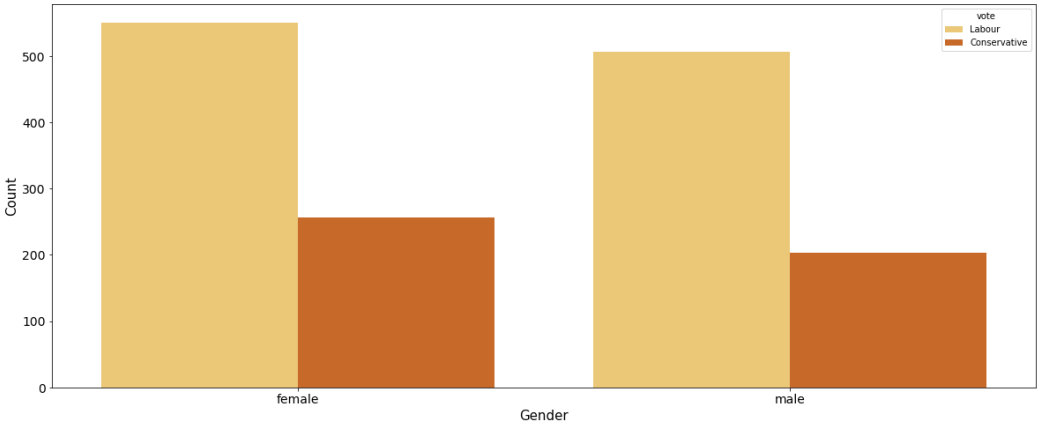


Figure 1.21: Vote to Parties.

- More of the males and females have given vote to the Labour party.

## Boxplot to check the Outliers:

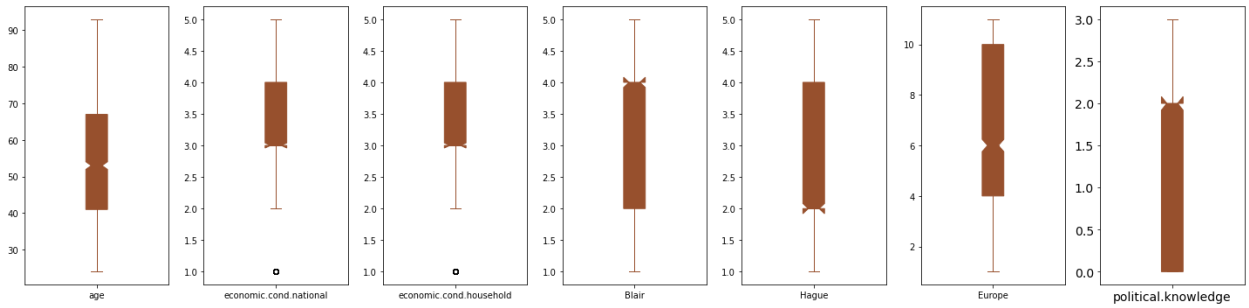


Figure 1.22: Boxplot to trace the outliers present in the dataset.

In the boxplot outliers have been found on economic condition national & household attributes, but the ratings can be 1, since the assessment of household/national economic conditions ranges from 1

to 5. Outliers are values which are mistakenly captured in the data, so all the values are correct and so treating these values would not be a wise decision from a business perspective.

## Checking for Correlations using Pair-plot and Heatmap:

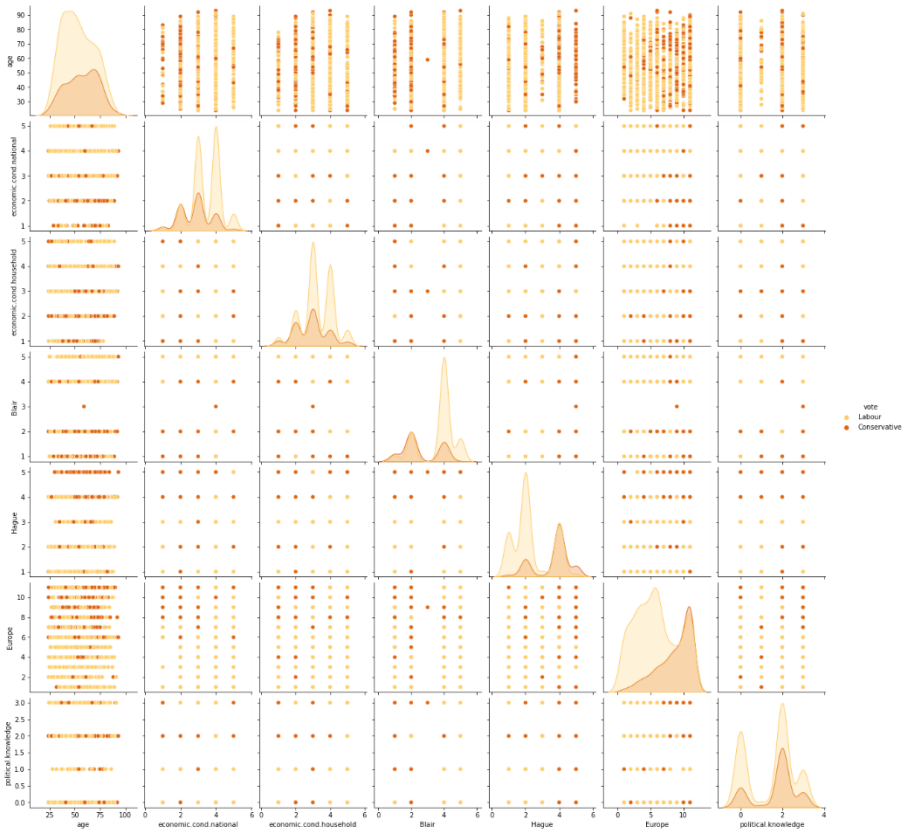


Figure 1.23: Pair-plot showing the relationship among the attributes of the dataset.

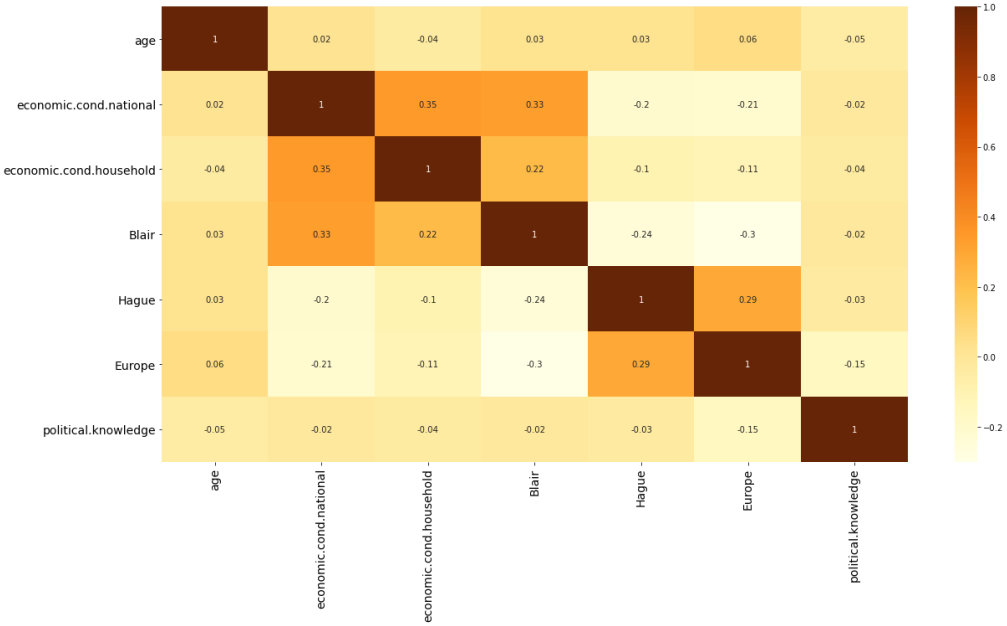


Figure 1.24: Heatmap representation of the correlation among the attributes of the dataset.

There is not a very strong correlation amongst any of the variables of the dataset.

## 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).

### Encode the data (having string values):

The dataset having 2 categorical variables 'gender' & 'vote' so one hot encoding with dropping of first column is right approach to avoid multicollenearity. Dummy variable is used to segregate between different sub-groups of the data for better regression analysis. The following code is used to get the dummy variable.

```
data_df =pd.get_dummies(data_df, columns=['gender'],drop_first=True)
```

Vote is the target attribute, therefore identifying a person who has voted for a Conservative or Labour party would be a better interpretation of the given model, and thus hot encoding was not necessary.

Since age group ranges from 20 to 100 and all other variables most of them are ordinal variables like rating ['economic.cond.national', 'economic.cond.household, 'Blair', 'Hague', 'Europe', 'political.knowledge'], So for better understanding and interpretation of the model binning of age has been performed as shown below.

```
data_df['age_bins'] = pd.cut(x=data_df['age'], bins=[20, 29, 39, 49,59,69,79,89,99],labels=['20s', '30s', '40s','50s','60s','70s','80s','90s'])
```

Persons over 20 years of age have been considered 20s, while those more than 30 years of age up to 39 have been considered 30s and so on. To transform those values to numerical values for modeling, they have been transformed into continuous values by ordinal encoding.

### Scaling:

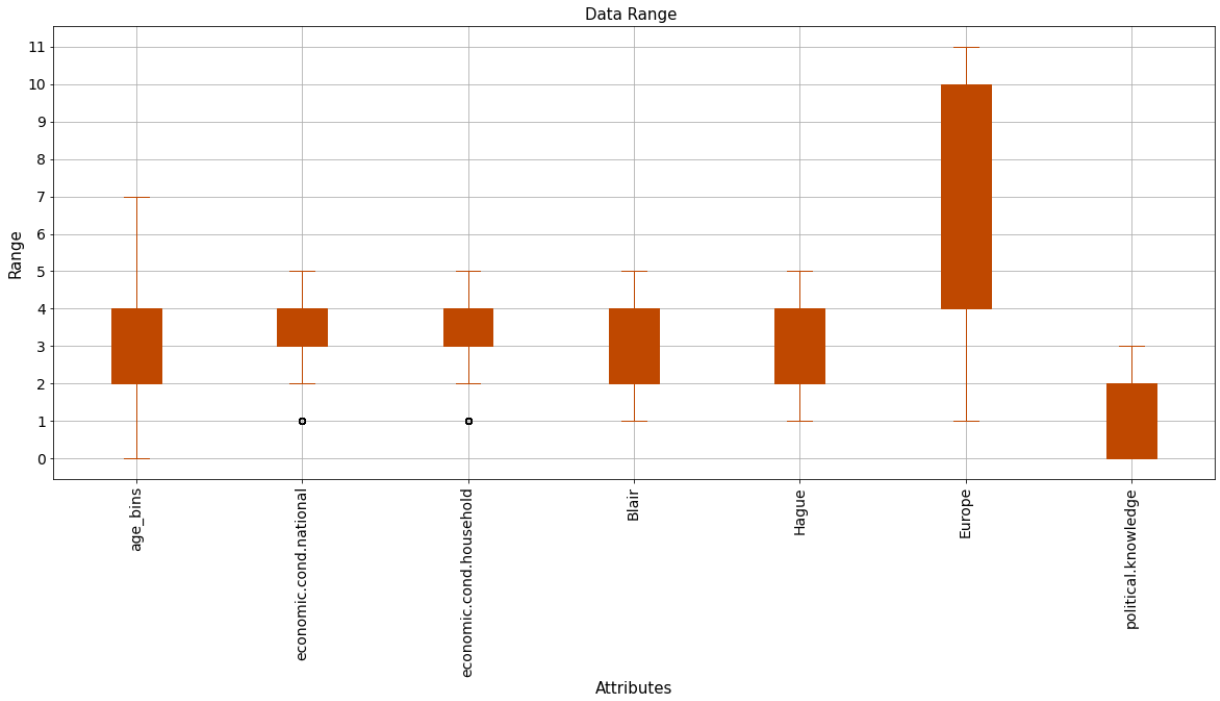


Figure 1.25: The attributes of the dataset range between 0 and 11.

According to Figure 1.25, data ranges are around 1 to 11 and most of them are ordinal, so scaling the ordinal variables makes no sense in this dataset.

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

Initially, the target variable (vote) was extracted from the dataset. In the dataset containing all of the independent attributes, two variables have been assigned, one (X) for the independent attribute and one (y) for the dependent attribute. By passing the parameters shown below, these two variables have been used to split the dataset (combination of dependent and independent attributes) into train and test data sets by using the python 'train_test_split' function.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=1)
```

The dimension of the train and test have been found as:

```
X_train:  (1061, 8)
X_test:  (456, 8)
y_train:  (1061,)
y_test:  (456,)
```

## 1.4 Apply Logistic Regression and LDA (linear discriminant analysis). Interpret the inferences of both models.

### Logistic Regression:

Logistic Regression has been applied by passing following parameters

```
model = LogisticRegression(solver='newton-cg', max_iter=10000,penalty='none',verbose=True, n_jobs=2)
model.fit(X_train, y_train);
```

newton-cg – Newton conjugate gradient method construction of the model.
The classification reports for both the train and test data have been obtained by passing above parameters as shown in Table 1.2a-b.

Table 1.2a: Logistic Regression-Train Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.74 | 0.64 | 0.68 | 307 |
| Labour | 0.86 | 0.91 | 0.88 | 754 |
| accuracy | | | **0.83** | 1061 |
| macro avg. | 0.80 | 0.77 | 0.78 | 1061 |
| weighted avg. | 0.83 | 0.83 | 0.83 | 1061 |

Table 1.2b: Logistic Regression-Test Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.76 | 0.73 | 0.74 | 153 |
| Labour | 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 |

The above model is giving comparatively low accuracy. Grid Search CV has to be used for the fine-tuning of the model.

### Linear Discriminant Analysis:

Linear Discriminant Analysis or LDA has been applied by passing following parameters

```
LDA_model= LinearDiscriminantAnalysis()
LDA_model.fit(X_train, y_train)
```

The default parameters that has been applied to LDA are solver='svd', shrinkage=None, priors=None, n_components=None, store_covariance=False, tol=0.0001, covariance_estimator=None

Since every model is having its best parameters as default and model building is an iterative process, so initially model has been constructed by the default parameters and tune the model by various iterations.

The classification reports for both the train and test data have been obtained by passing above parameters as shown in Table 1.3a-b.

Table 1.3a: Linear Discriminant Analysis -Train Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.74 | 0.65 | 0.69 | 307 |
| Labour | 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 |

Table 1.3b: Linear Discriminant Analysis -Test Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.78 | 0.73 | 0.75 | 153 |
| Labour | 0.87 | 0.89 | 0.88 | 303 |
| accuracy | | | **0.84** | 456 |
| macro avg. | 0.82 | 0.81 | 0.81 | 456 |
| weighted avg. | 0.84 | 0.84 | 0.84 | 456 |

Accuracy in both the training and test data has been found comparable thus fine tuning of the model by iterating various cut off probabilities and also use of Grid Search CV for improve the model.

After various iterations the following cut-off probabilities have been derived as shown in Table 1.4.

Table 1.4: The threshold probability

| Threshold | Accuracy | F-1 Score | Recall |
|---|---|---|---|
| 0.1 | 0.761 | 0.854 | 0.987 |
| 0.2 | 0.791 | 0.868 | 0.968 |
| 0.3 | 0.812 | 0.878 | 0.952 |
| **0.4** | **0.832** | **0.888** | **0.939** |
| 0.5 | 0.833 | 0.885 | 0.907 |
| 0.6 | 0.826 | 0.876 | 0.87 |
| 0.7 | 0.833 | 0.877 | 0.838 |

| | | | |
|---|---|---|---|
| 0.8 | 0.79 | 0.835 | 0.751 |
| 0.9 | 0.697 | 0.737 | 0.599 |

From above table at threshold probability 0.4 the accuracy & F1 Scores have been found comparatively high. So 0.4 has been considered as the cut-off probabilities in the model.

Classification reports of the default and custom cut-off test data have been described in Table 1.5a-b.

Table 1.5a: Classification Report of the default cut-off test data

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.78 | 0.73 | 0.75 | 153 |
| Labour | 0.87 | 0.89 | 0.88 | 303 |
| accuracy | | | **0.84** | 456 |
| macro avg. | 0.82 | 0.81 | 0.81 | 456 |
| weighted avg. | 0.84 | 0.84 | 0.84 | 456 |

Table 1.5b: Classification Report of the custom cut-off test data

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.79 | 0.73 | 0.72 | 153 |
| Labour | 0.84 | 0.89 | 0.87 | 303 |
| accuracy | | | **0.83** | 456 |
| macro avg. | 0.82 | 0.81 | 0.80 | 456 |
| weighted avg. | 0.82 | 0.84 | 0.82 | 456 |

The two algorithms have been summarized in Table 1.6 from the classification tables. In those tables Labour has been considered as positive class.

Table 1.6: Summary of the classification tables for Logistic Regression and Linear Discriminant Analysis

| | Train Recall | Test Recall | Accuracy Train | Accuracy Test |
|---|---|---|---|---|
| Logistic Regression | 0.91 | 0.88 | 0.83 | 0.83 |
| Linear Discriminant Analysis | 0.91 | 0.89 | 0.83 | 0.84 |

As per the Table 1.6, recall and accuracy scores are slightly better in case of Linear Discriminant Analysis.

From the Table 1.5a-b, it has been clearly observed that accuracy has a fraction of increment which is not good so Grid search CV has to be incorporated for further tuning.

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

### KNN Model:

K-Nearest Neighbor or KNN algorithm has been applied by passing following parameters.

```
KNN_model=KNeighborsClassifier(n_neighbors=7)
KNN_model.fit(X_train,y_train)
```

The classification reports for both the train and test data have been obtained by passing above parameters as shown in Table 1.7a-b.

Table 1.7a: KNN algorithm -Train Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.78 | 0.70 | 0.74 | 307 |
| Labour | 0.88 | 0.92 | 0.90 | 754 |
| accuracy | | | **0.86** | 1061 |
| macro avg. | 0.83 | 0.81 | 0.82 | 1061 |
| weighted avg. | 0.85 | 0.86 | 0.85 | 1061 |

Table 1.7b: KNN algorithm -Test Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.74 | 0.64 | 0.69 | 153 |
| Labour | 0.83 | 0.89 | 0.86 | 303 |
| accuracy | | | **0.80** | 456 |
| macro avg. | 0.79 | 0.76 | 0.77 | 456 |
| weighted avg. | 0.80 | 0.80 | 0.80 | 456 |

Since every model is having its best parameters as default and model building is an iterative process, so initially model has been constructed by the default parameters and tune the model by various iterations.

In the above model, 7 nearest neighbors has been used and model is giving comparatively low accuracy. Changing of nearest neighbors has to be used for fine tuning of the model.

### Naïve Bayes Model:

Naïve Bayes algorithm has been applied by passing following parameters. The default parameters that has been applied to Naïve Bayes algorithm are priors=None, var_smoothing=1e-09

```
NB_model = GaussianNB()

NB_model.fit(X_train, y_train)
```

The classification reports for both the train and test data have been obtained by passing above parameters as shown in Table 1.8a-b.

Table 1.8a: Naïve Bayes algorithm -Train Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.73 | 0.69 | 0.71 | 307 |
| Labour | 0.88 | 0.89 | 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 |

Table 1.8b: Naïve Bayes algorithm -Test Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.74 | 0.73 | 0.73 | 153 |
| Labour | 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 |

Since every model is having its best parameters as default and model building is an iterative process, so initially model has been constructed by the default parameters and tune the model by various iterations.

The confusion matrix for both the Train and Test data have been described in Figure 1.27. The algorithm correctly predicted 213 votes for conservative party and 674 votes for Labour party and 174 predictions are wrong in Train set, and similarly 111 votes correctly predicted for conservative party and 263 votes for Labour party and 82 predictions are wrong in Test set.

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

### Model Tuning: Logistic Regression

The logistic regression model has been tuned by applying Grid search CV by passing following parameters

```
grid={'penalty':['l2','l1'], 'solver':['sag','lbfgs'], 'tol':[0.0001,0.00001]}
```

*l1*: Lasso regression penalty & *Lbfgs:* Stands for Limited-memory Broyden–Fletcher–Goldfarb–Shanno. It approximates the second derivative matrix updates with gradient evaluations. It stores only the last few updates.

```
GridSearchCV(cv=3, estimator=LogisticRegression(max_iter=10000, n_jobs=2), n_jobs=-1, param_grid={'penalty': ['l2', 'l1'], 'solver': ['sag', 'lbfgs'], 'tol': [0.0001, 1e-05]}, scoring='accuracy')
```

The best parameters by using Grid search CV for Logistic Regression have been found as, {'penalty': 'l2', 'solver': 'sag', 'tol': 0.0001}

The best parameters have been found are *l2*: Ridge regression is applied when the issue of multicollenearity occurs*, solver* is sag (Stochastic Average Gradient) in Stochastic Gradient Descent, a few samples are selected randomly instead of the whole data set for each iteration & *tolerance* is 0.001.

### Model Tuning: Linear Discriminant Analysis

The linear discriminant analysis model has been tuned by applying Grid search CV by passing following parameters.

```
grid={'solver':['lsqr','eigen'],  'n_components':[1,7,2]}
```

```
GridSearchCV(cv=5,estimator=LinearDiscriminantAnalysis(),n_jobs=-1,
                   param_grid={'n_components': [1, 7, 2], 'solver': ['lsqr', 'eigen']})
```

The best parameters by using Grid search CV for Linear Discriminant Analysis have been found as, {'n_components': 1, 'solver': 'lsqr'}

Lsqr: It is the *solver* which uses least square method for discriminate between two classes.

### Model Tuning: K-Nearest Neighbor

The success of KNN model depends upon by choosing the best K- value where misclassification error should be minimal. The errors have been calculated for various k values and plotted them as shown in Figure 1.26.

The misclassification error is found minimal on K=17 value. This means best accuracy of the model has been constructed by considering 17 nearest neighbors. In this context, the misclassification error or MCE is equal to 1 - Test accuracy score.

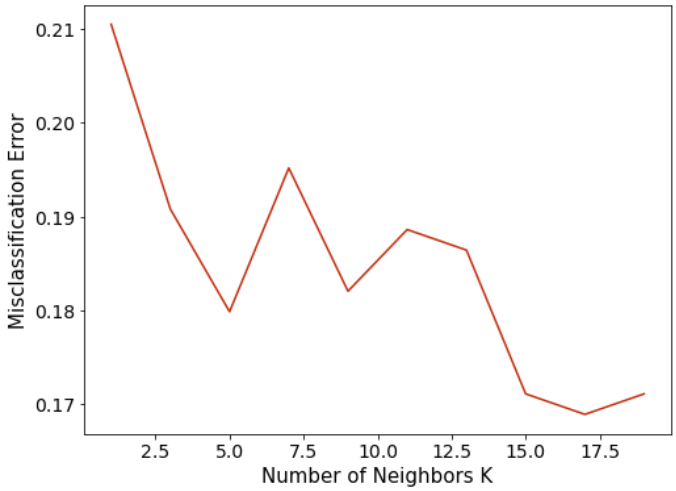


Figure 1.26: Calculated MCE for each model with neighbors = 1, 3, 5...19.

### Model Tuning: Naïve Bays Model

### Naïve Bays model has been tuned by bagging.

Models which have performed approximately similar on the Train and Test data set and application of SMOTE to the same to check if the performance improves or not. SMOTE thus has been applied on Naïve bays and KNN algorithm (Table 1.9). It has been observed that there is no significant effect of SMOTE on any model and there is significant difference between accuracy for KNN since SMOTE is a technique generally applied if minority class is below 5%.

Table 1.9: Performance of SMOTE on Naïve bays and KNN

| | Accuracy Train | Accuracy Test |
|---|---|---|
| Naive-Bayes SMOTE | 0.834218 | 0.802632 |
| KNN SMOTE | 0.888594 | 0.804825 |

### Bagging (Random Forest should be applied for Bagging):

Random forest Model has been applied for introducing Bootstrapped Aggregating by passing below parameters.

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

The classification reports for both the train and test data have been obtained by passing above parameters as shown in Table 1.10a-b.

Table 1.10a: Bagging -Random forest Model-Train Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.98 | 0.89 | 0.93 | 307 |
| Labour | 0.96 | 0.99 | 0.97 | 754 |
| accuracy | | | **0.96** | 1061 |
| macro avg. | 0.80 | 0.79 | 0.95 | 1061 |
| weighted avg. | 0.83 | 0.84 | 0.96 | 1061 |

Table 1.10b: Bagging -Random forest Model-Test Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.79 | 0.67 | 0.73 | 153 |
| Labour | 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 |

With random forest Bagging that Train data Accuracy has been found 96% and test data accuracy is 83%. So there will be overfitting. Because the model is performing better Train set but comparatively less performing in Test Data.

Boosting:

Ada Boost and Gradient Boost have been used for modelling. The AdaBoostClassifier has been used for Adaboost whiel Grdient Boost Classifier has been for Gredient boost by passing below parameters.

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

The classification reports for both the train and test data have been obtained by passing above parameters as shown in Table 1.11a-b and Table 1.12a-d.

Table 1.11a: Ada Boost Model-Train Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.75 | 0.68 | 0.71 | 307 |
| Labour | 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 |

Table 1.11b: Ada Boost Model-Test Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.76 | 0.69 | 0.72 | 153 |
| Labour | 0.85 | 0.89 | 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 |

Table 1.12a: Gradient Boost Model-Train Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.83 | 0.77 | 0.80 | 307 |
| Labour | 0.91 | 0.93 | 0.92 | 754 |
| accuracy | | | **0.89** | 1061 |
| macro avg. | 0.87 | 0.85 | 0.86 | 1061 |
| weighted avg. | 0.89 | 0.89 | 0.89 | 1061 |

Table 1.12b: Gradient Boost Model-Test Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.78 | 0.69 | 0.73 | 153 |
| Labour | 0.85 | 0.90 | 0.88 | 303 |
| accuracy | | | **0.83** | 456 |
| macro avg. | 0.82 | 0.80 | 0.80 | 456 |
| weighted avg. | 0.83 | 0.83 | 0.83 | 456 |

After performing all of the above algorithm a comparative chart (Table 1.13) has been prepared.

Table 1.13: Comparison of different algorithm

| | Train Recall | Test Recall | Accuracy Train | Accuracy Test |
|---|---|---|---|---|
| Naive-Bayes | 0.893899 | 0.867987 | 0.836004 | 0.820175 |
| Logistic Regression | 0.909814 | 0.884488 | 0.830349 | 0.83114 |
| LDA | 0.907162 | 0.894389 | 0.833176 | 0.837719 |
| ADA Boost | 0.905836 | 0.887789 | 0.840716 | 0.822368 |
| Gradient Boost | 0.933687 | 0.90429 | 0.886899 | 0.83114 |
| KNN | 0.901857 | 0.907591 | 0.838831 | 0.83114 |
| Decision Tree | 0.98939 | 0.815182 | 0.99246 | 0.758772 |
| Random Forest | 0.996021 | 0.89769 | 0.99246 | 0.811404 |
| Bagging | 0.992042 | 0.907591 | 0.961357 | 0.828947 |

From above table, the accuracy has been observed for Boosting & Bagging with Random forest and Naïve Bays. Bagging model has been found as an over fitted model and Gradient Boost has been observed performing well in this case.

## 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 present study accuracy has been selected as the performance parameter.

### Logistic Regression Performance Matrices:

The classification reports for both the train and test data have been obtained by passing above parameters as shown in Table 1.14a-b.

Table 1.14a: Tuned Logistic Regression Model-Train Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.74 | 0.64 | 0.68 | 307 |
| Labour | 0.86 | 0.91 | 0.88 | 754 |
| accuracy | | | **0.83** | 1061 |
| macro avg. | 0.80 | 0.77 | 0.78 | 1061 |
| weighted avg. | 0.83 | 0.83 | 0.83 | 1061 |

Table 1.14b: Tuned Logistic Regression Model-Test Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.76 | 0.73 | 0.74 | 153 |
| Labour | 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 |

Logistic Regression model accuracy is comparable i.e. 83% for both Training and Test data, so this model can be a good model.

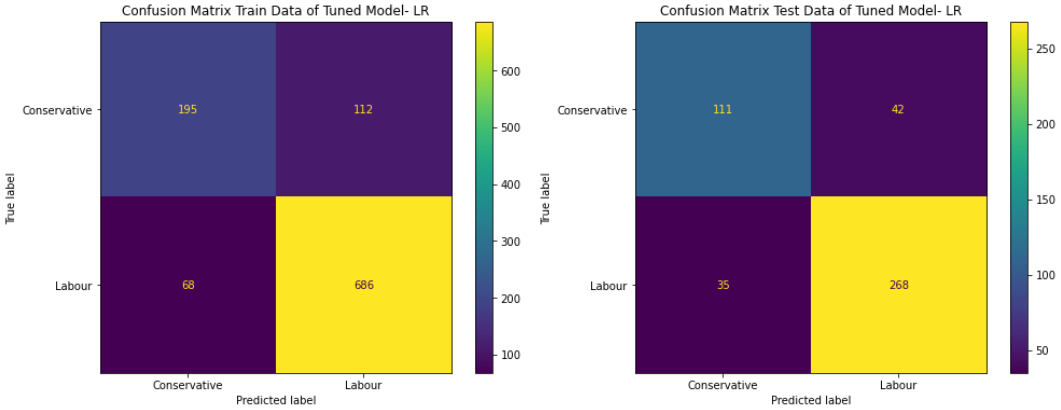


Figure 1.27: Confusion matrices for training and test data for Logistic Regression model

The confusion matrix obtained from tuned Logistic Regression model for both the Train and Test data have been described in Figure 1.27. The algorithm correctly predicted 195 votes for conservative party and 686 votes for Labour party and 180 predictions are wrong in Train set, and similarly 111 votes correctly predicted for conservative party and 268 votes for Labour party and 77 predictions are wrong in Test set.

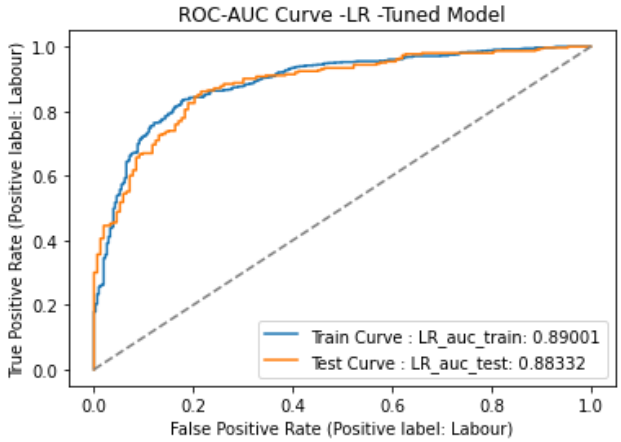


Figure 1.28: ROC for training and test data for Logistic Regression model

The area under curve i.e. AUC has been found from the above ROC AUC Curve (Figure 1.28) for Train is 89% and for Test it is 88%. Hence, 88% of the time the tuned Logistic Regression model performs well in Test data.

### Linear Discriminant Analysis Performance Matrices:

The classification reports for both the train and test data have been obtained by passing above parameters as shown in Table 1.15a-b.

Table 1.15a: Tuned Linear Discriminant Analysis Model-Train Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.74 | 0.65 | 0.69 | 307 |

| | | | | |
|---|---|---|---|---|
| Labour | 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 |

Table 1.15b: Tuned Linear Discriminant Analysis Model-Test Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.78 | 0.73 | 0.75 | 153 |
| Labour | 0.87 | 0.89 | 0.88 | 303 |
| accuracy | | | **0.84** | 456 |
| macro avg. | 0.82 | 0.81 | 0.81 | 456 |
| weighted avg. | 0.84 | 0.84 | 0.84 | 456 |

Linear Discriminant Analysis model accuracy is comparable i.e. 83% for Training and 84% for Test data, so this model can be further improved by iterative methods.

The confusion matrix obtained from tuned Linear Discriminant Analysis model for both the Train and Test data have been described in Figure 1.29. The algorithm correctly predicted 200 votes for conservative party and 684 votes for Labour party and 177 predictions are wrong in Train set, and similarly 111 votes correctly predicted for conservative party and 271 votes for Labour party and 74 predictions are wrong in Test set.

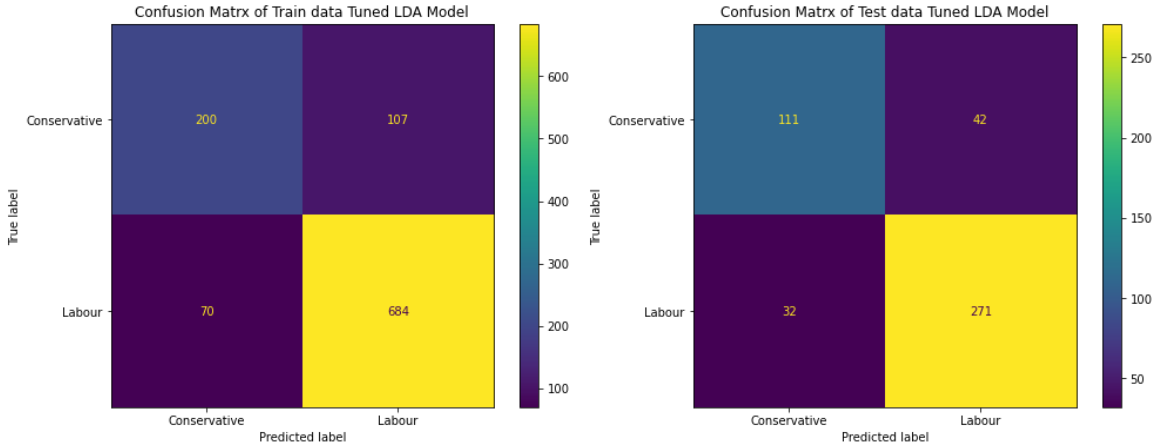


Figure 1.29: Confusion matrices for training and test data for Linear Discriminant Analysis model

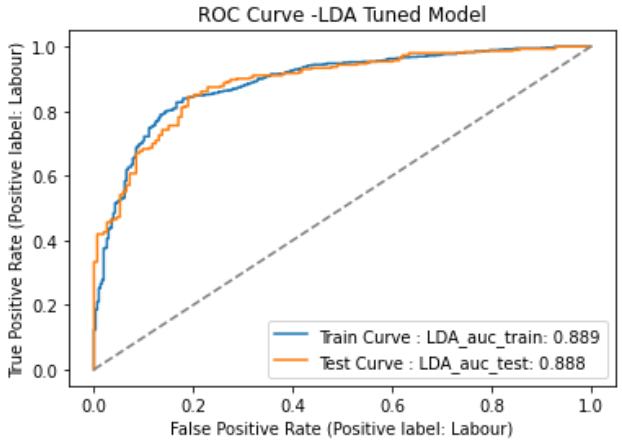


Figure 1.30: ROC for training and test data for Linear Discriminant Analysis model

AUC has been found from the above ROC AUC Curve (Figure 1.30) for Train and Test data is 88%. Hence, model shall be performing well.

### K- Nearest Neighbour Performance Matrices:

The classification reports for both the train and test data have been obtained by passing above parameters as shown in Table 1.16a-b.

Table 1.16a: Tuned K- Nearest Neighbour Model-Train Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.74 | 0.68 | 0.71 | 307 |
| Labour | 0.88 | 0.90 | 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 |

Table 1.16b: Tuned K- Nearest Neighbour Model-Test Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.79 | 0.68 | 0.73 | 153 |
| Labour | 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 | 0.843 | 456 |

K- Nearest Neighbour model accuracy is comparable i.e. 84% for Training and 83% for Test data, so this model can be further improved by iterative methods.

The confusion matrix obtained from tuned K- Nearest Neighbour model for both the Train and Test data have been described in Figure 1.31. The algorithm correctly predicted 210 votes for conservative party and 680 votes for Labour party and 171 predictions are wrong in Train set, and similarly 104 votes correctly predicted for conservative party and 275 votes for Labour party and 77 predictions are wrong in Test set.

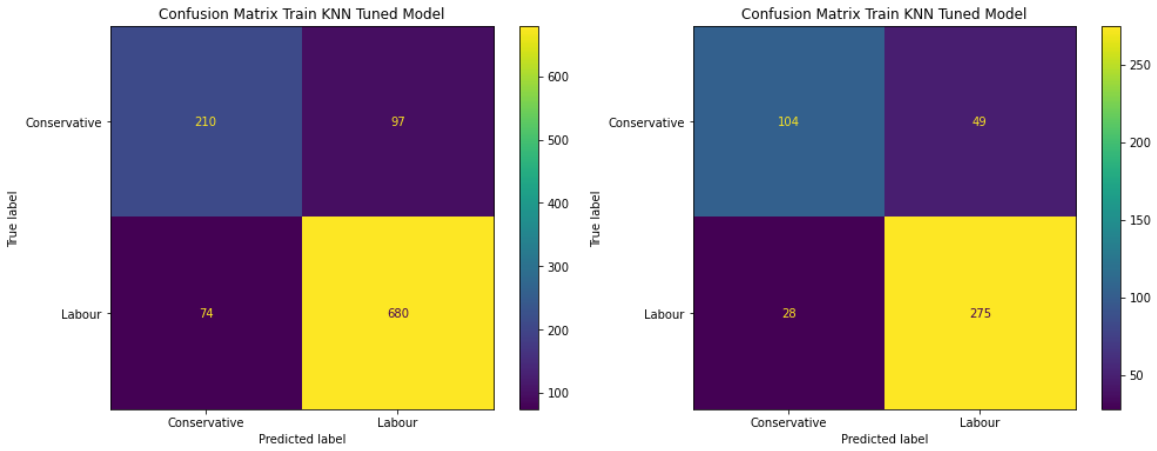


Figure 1.31: Confusion matrices for training and test data for K- Nearest Neighbour model

The area under curve i.e. AUC has been found from the above ROC AUC Curve (Figure 1.32) for Train is 90% and for Test it is 89%. Hence, 89% of the time the tuned KNN model performs well in Test data.

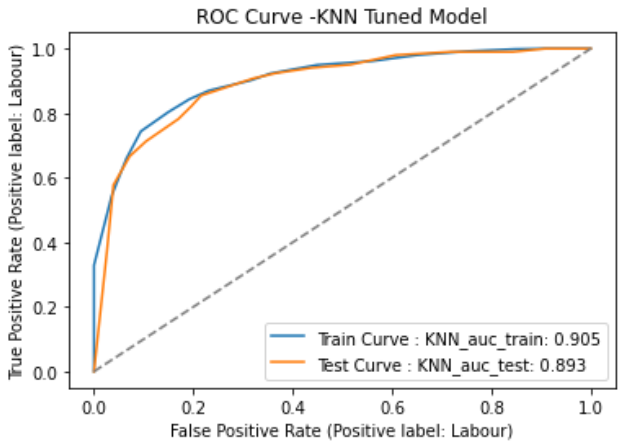


Figure 1.32: ROC for training and test data for K- Nearest Neighbour model

### Naïve Bays Performance Matrices:

The classification reports for both the train and test data have been obtained by passing above parameters as shown in Table 1.17a-b.

Table 1.17a: Tuned Naïve Bays Model-Train Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.73 | 0.69 | 0.71 | 307 |
| Labour | 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 |

Table 1.17b: Tuned Naïve Bays Model-Test Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.74 | 0.73 | 0.73 | 153 |
| Labour | 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 |

Naïve Bays model accuracy is comparable i.e. 84% for Training and 82% for Test data Hence, 82% of the time the tuned Naïve Bays model performs well in Test data.

The confusion matrix obtained from tuned Naïve Bays model for both the Train and Test data have been described in Figure 1.33. The algorithm correctly predicted 211 votes for conservative party and 675 votes for Labour party and 175 predictions are wrong in Train set, and similarly 111 votes correctly predicted for conservative party and 264 votes for Labour party and 81 predictions are wrong in Test set.

The area under curve i.e. AUC has been found from the above ROC AUC Curve (Figure 1.34) for Train is 88% and for Test it is 87%. Hence, 87% of the time the tuned Naïve Bays model performs well in Test data.

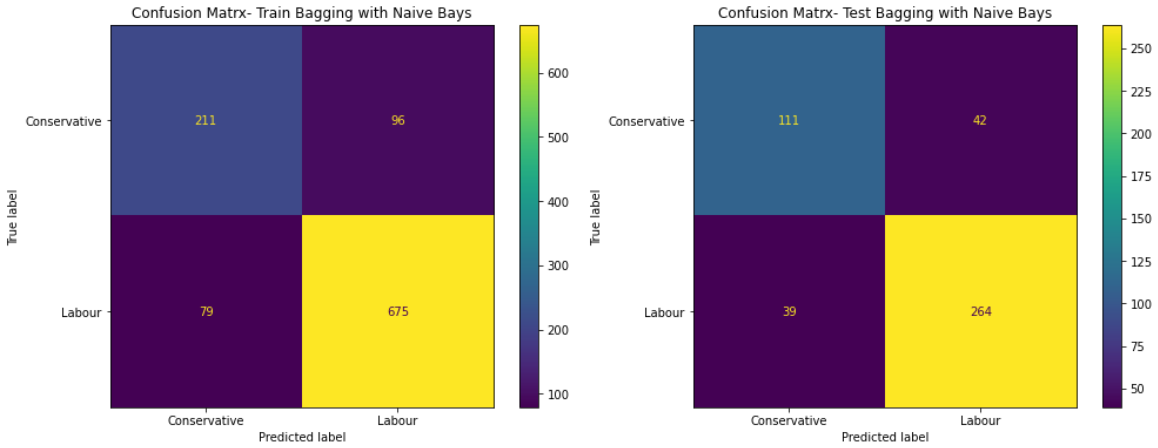


Figure 1.33: Confusion matrices for training and test data for Naïve Bays model

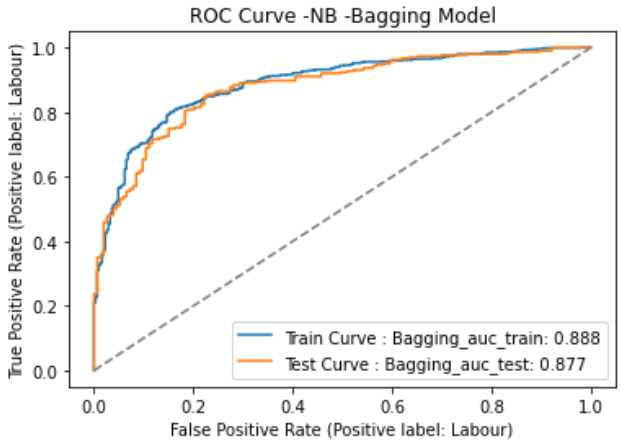


Figure 1.34: ROC for training and test data for Naïve Bays model

### Ada Boost Performance Matrices:

The classification reports for both the train and test data have been obtained by passing above parameters as shown in Table 1.18a-b.

Table 1.18a: Tuned Ada Boost Model-Train Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.75 | 0.68 | 0.71 | 307 |
| Labour | 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 |

Table 1.18b: Tuned Ada Boost Model-Test Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.76 | 0.69 | 0.72 | 153 |
| Labour | 0.85 | 0.89 | 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 |

Ada Boost model accuracy is comparable i.e. 84% for Training and 82% for Test data. Hence, 82% of the time the tuned Ada Boost model performs well in Test data.

The confusion matrix obtained from tuned Ada Boost model for both the Train and Test data have been described in Figure 1.35. The algorithm correctly predicted 209 votes for conservative party and 683 votes for Labour party and 169 predictions are wrong in Train set, and similarly 106 votes correctly predicted for conservative party and 269 votes for Labour party and 81 predictions are wrong in Test set.

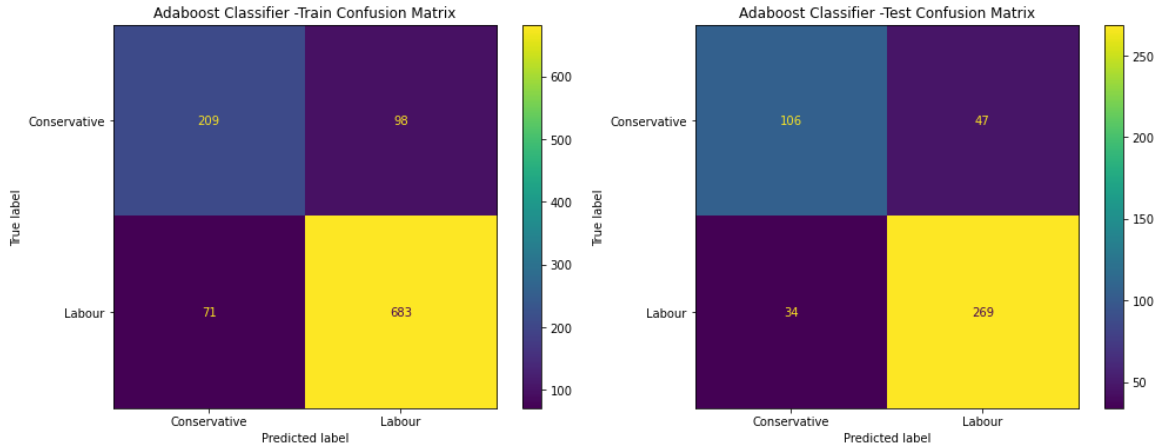


Figure 1.35: Confusion matrices for training and test data for Ada Boost model

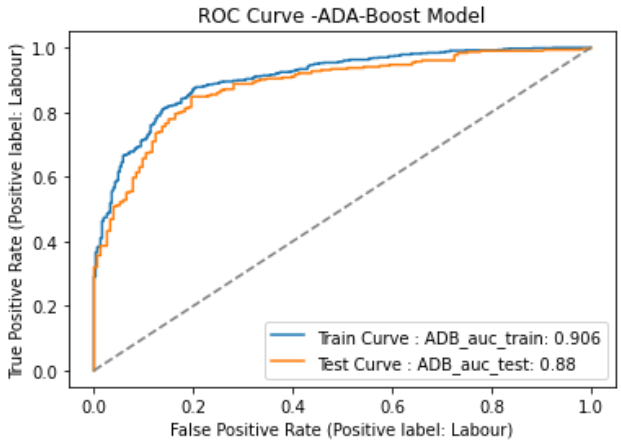


Figure 1.36: ROC for training and test data for Ada Boost model

The area under curve i.e. AUC has been found from the above ROC AUC Curve (Figure 1.36) for Train is 90% and for Test it is 88%. Hence, 88% of the time the tuned Ada Boost model performs well in Test data.

### Gradient Boost Performance Matrices:

The classification reports for both the train and test data have been obtained by passing above parameters as shown in Table 1.19a-b.

Table 1.19a: Gradient Boost Model-Train Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.83 | 0.77 | 0.80 | 307 |
| Labour | 0.91 | 0.93 | 0.92 | 754 |

| | | | | |
|---|---|---|---|---|
| accuracy | | | **0.89** | 1061 |
| macro avg. | 0.87 | 0.85 | 0.86 | 1061 |
| weighted avg. | 0.89 | 0.89 | 0.89 | 1061 |

Table 1.19b: Gradient Boost Model-Test Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.78 | 0.69 | 0.73 | 153 |
| Labour | 0.85 | 0.90 | 0.88 | 303 |
| accuracy | | | **0.83** | 456 |
| macro avg. | 0.82 | 0.80 | 0.80 | 456 |
| weighted avg. | 0.83 | 0.83 | 0.83 | 456 |

Gradient Boost model accuracy is comparable i.e. 89% for Training and 83% for Test data. Hence, 83% of the time the tuned Gradient Boost model performs well in Test data.

The confusion matrix obtained from tuned Gradient Boost model for both the Train and Test data have been described in Figure 1.37. The algorithm correctly predicted 237 votes for conservative party and 704 votes for Labour party and 120 predictions are wrong in Train set, and similarly 105 votes correctly predicted for conservative party and 274 votes for Labour party and 77 predictions are wrong in Test set.

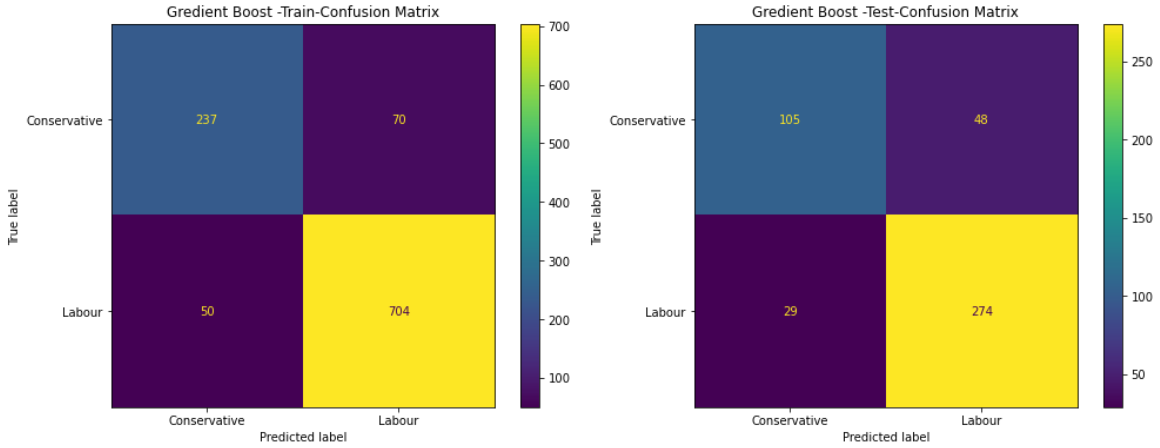


Figure 1.37: Confusion matrices for training and test data for Gradient Boost model

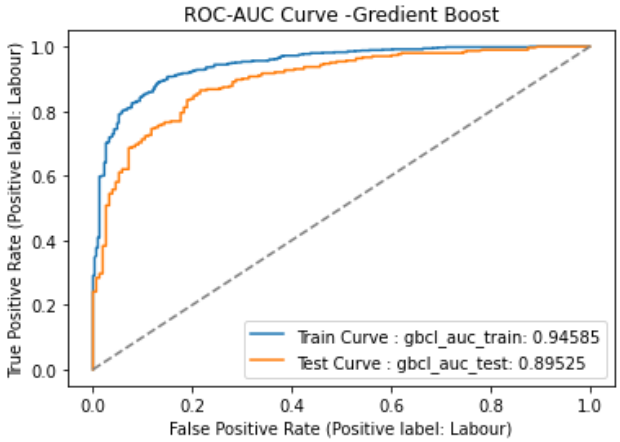


Figure 1.38: ROC for training and test data for Gradient Boost model

The area under curve i.e. AUC has been found from the above ROC AUC Curve (Figure 1.38) for Train is 94% and for Test it is 89%. Hence, 89% of the time the tuned Gradient Boost model performs well in Test data.

### Bagging Performance Matrices:

The classification reports for both the train and test data have been obtained by passing above parameters as shown in Table 1.20a-b.

Table 1.20a: Tuned Bagging Model-Train Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.98 | 0.89 | 0.93 | 307 |
| Labour | 0.96 | 0.99 | 0.97 | 754 |
| accuracy | | | **0.96** | 1061 |
| macro avg. | 0.80 | 0.79 | 0.95 | 1061 |
| weighted avg. | 0.83 | 0.84 | 0.96 | 1061 |

Table 1.20b: Tuned Bagging Model-Test Data classification report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Conservative | 0.79 | 0.67 | 0.73 | 153 |
| Labour | 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 |

The tuned Bagging model accuracy is comparable i.e. 96% for Training and 83 % for Test data. There might be overfitting. The model is performing better in Train set but comparatively less performing in Test data.

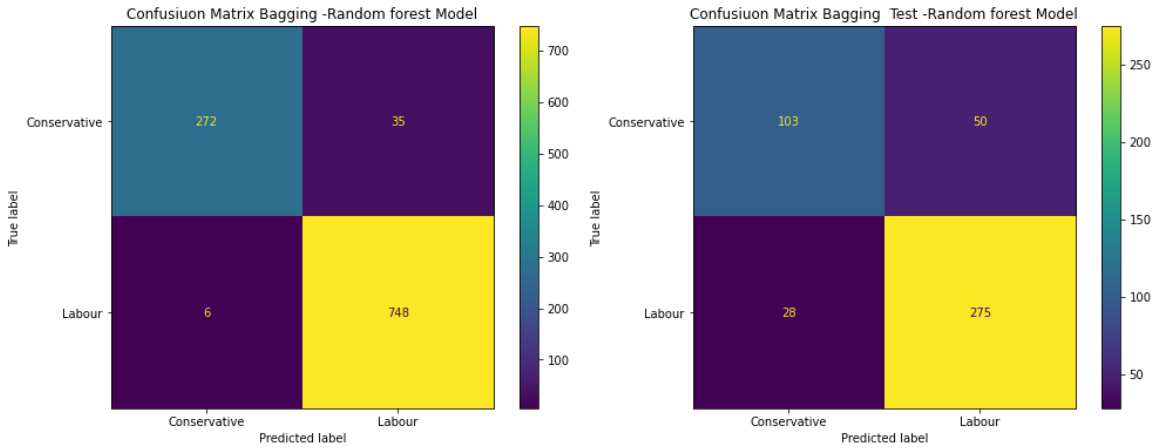


Figure 1.39: Confusion matrices for training and test data for tuned Bagging model

The confusion matrix obtained from tuned Bagging model for both the Train and Test data have been described in Figure 1.39. The algorithm correctly predicted 272 votes for conservative party and 748 votes for Labour party and 41 predictions are wrong in Train set, and similarly 103 votes correctly predicted for conservative party and 275 votes for Labour party and 78 predictions are wrong in Test set.

The area under curve i.e. AUC has been found from the above ROC AUC Curve (Figure 1.40) for Train is 99% and for Test it is 89 %. Hence, 88% of the time the tuned Bagging model performs well in Test data.

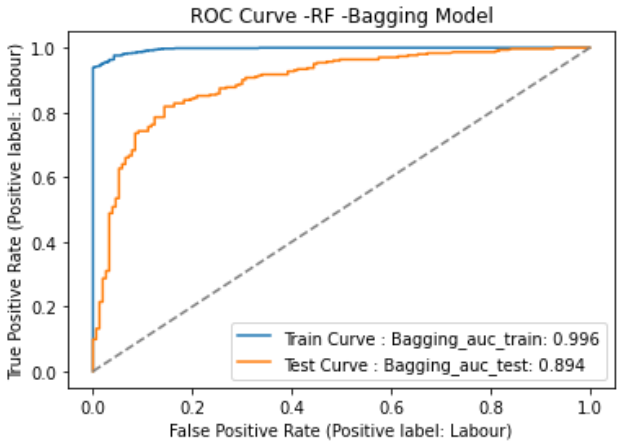


Figure 1.40: ROC for training and test data for tuned Bagging model

### Final Model:

After building all the models performance metrics have been summarized in the Table 1.45 and Figure 1.21.

Table 1.21: Performance metrics

| | Train Recall | Test Recall | Accuracy Train | Accuracy Test |
|---|---|---|---|---|
| Naive-Bayes SMOTE | 0.830239 | 0.811881 | 0.834218 | 0.802632 |
| Logistic Regression | 0.909814 | 0.884488 | 0.830349 | 0.83114 |
| LDA | 0.907162 | 0.894389 | 0.833176 | 0.837719 |
| ADA Boost | 0.905836 | 0.887789 | 0.840716 | 0.822368 |
| Gradient Boost | 0.933687 | 0.90429 | 0.886899 | 0.83114 |
| KNN SMOTE | 0.831565 | 0.815182 | 0.888594 | 0.804825 |
| Decision Tree | 0.98939 | 0.815182 | 0.99246 | 0.758772 |
| Random Forest | 0.996021 | 0.89769 | 0.99246 | 0.811404 |
| Bagging | 0.992042 | 0.907591 | 0.961357 | 0.828947 |

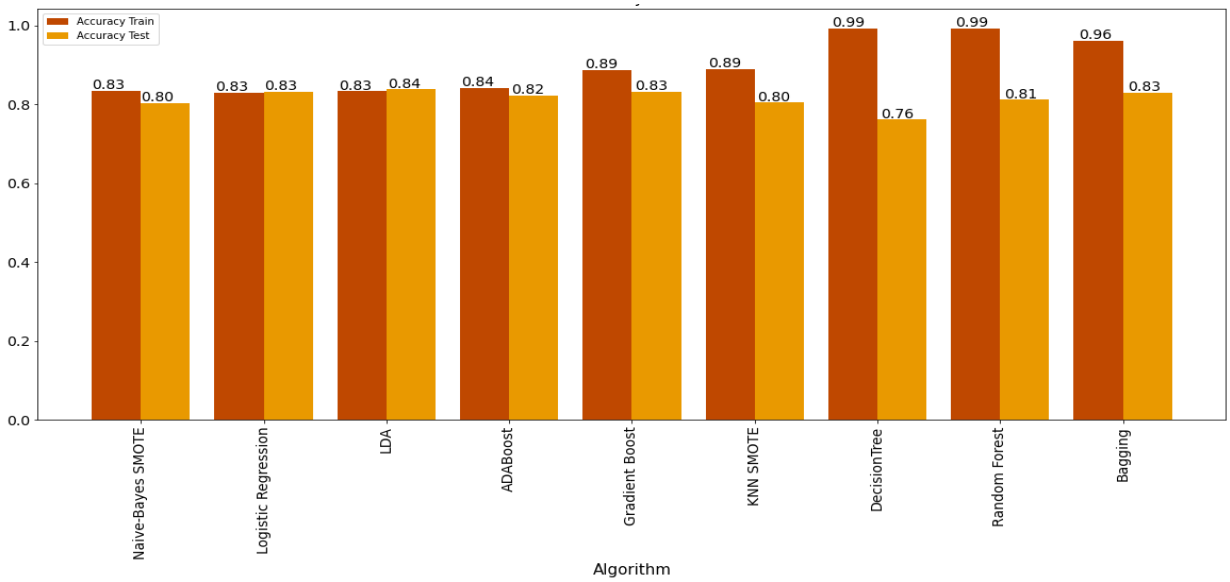


Figure 1.41: Comparison among the accuracy of different models

According to Table 1.45, Naïve Bays SMOTE and KNN SMOTE Test accuracy is 80%, LR, Gradient boost, and Bagging Test accuracy is 83%, LDA Test accuracy is 84%, ADA Boost Test accuracy 82%, Random Forest Test accuracy is 80%, and Decision Tree Test accuracy is 76%.

Similarly, according to Bar Graph (Figure 1.41), LDA model is giving comparatively higher Test Accuracy. In addition to the Training and Test accuracy are fitted perfectly, thus LDA model is best optimized model.

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

The purpose of this problem is to predict which party a voter will vote for based on the given information. Different types of models have been used to solve this problem, such as Logistic Regression, Linear Discriminant Analysis, k-Nearest Neighbor and Naïve Bays models. Based on the results obtained, LDA is performing better than other models in terms of accuracy.

The insights of the study are as follows:

1. A low correlation between the variables has been noted, which is a positive for the model
2. The 30s to 70s age group votes the most.
3. Young and old voters are significantly less likely to vote.
4. Voting for Labour party has been significantly more popular among women
5. People with Eurosceptic attitudes voted for Labour.
6. In the election, Labour got more votes than Conservatives.

Recommendations:

1. Collect more information, such as ratings on their previous performances.
2. Religion of the respondents, etc., to gain a better understanding.
3. It may be possible to reduce actual survey costs through online surveys so as to collect more data.
4. The company can collect ratings on the leadership's attitude towards current issues.

# Problem 2: Inaugural Corpora

The objective of the work is on the inaugural corpora on 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

The libraries have been used to solve the problem are as follows:

```
import nltk

import re

import wordcloud

from nltk.corpus import inaugural
```

After downloading the speeches, a data frame has been formed to analyze the speeches. The code has been used is as follows:

```
index = ['1941-Roosevelt','1961-Kennedy','1973-Nixon']

inaugural_speech=pd.DataFrame({'president':['Roosevelt(1941)','Kennedy(1961)','Nixon(1973)'],
'text':[inaugural.raw('1941Roosevelt.txt'),inaugural.raw('1961-Kennedy.txt'),inaugural.raw('1973-Nixon.txt')]},index=index)
```

The newly constructed data frame (Figure 2.1) has been named in this work as 'inaugural_speech'.

| | president | text |
|---|---|---|
| **1941-Roosevelt** | Roosevelt(1941) | On each national day of inauguration since 178... |
| **1961-Kennedy** | Kennedy(1961) | Vice President Johnson, Mr. Speaker, Mr. Chief... |
| **1973-Nixon** | Nixon(1973) | Mr. Vice President, Mr. Speaker, Mr. Chief Jus... |

Figure 2.1: The newly constructed data frame, 'inaugural_speech'

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

### Number of characters:

The code 'inaugural_speech['text'].str.len()' has been used to get the total counts of the characters used in the three speeches(Figure 2.2). The space between the two words also considered here.

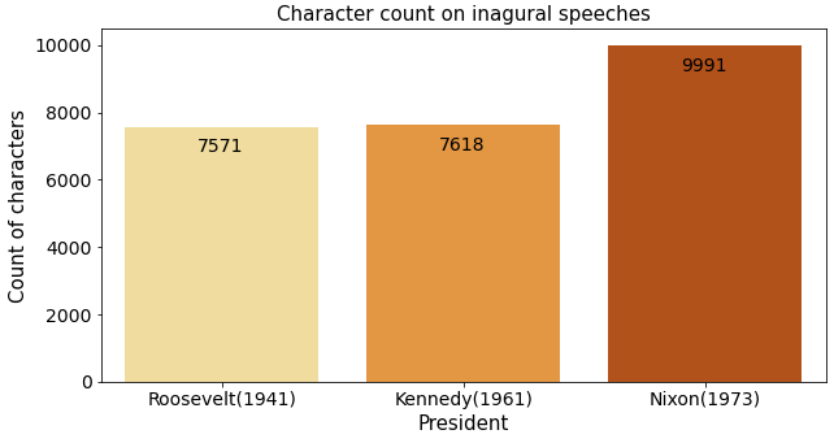


Figure 2.2: Character counts on inaugural speeches given by three Presidents

The numbers in the Figure 2.2, depicts the counts of character in the speech which are 7571 of President Roosevelt, 7618 of President Kennedy, and 9991 of President Nixon.

### Number of words:

The number of words in the speech are, 1323 of President Roosevelt, 1364 of President Kennedy, and 1769 of President Nixon has been depicted in Figure 2.3.

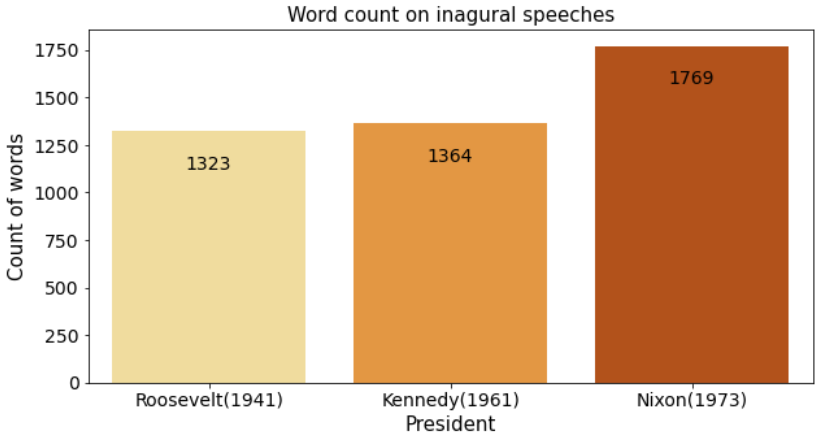


Figure 2.3: Word counts on inaugural speeches given by three Presidents

### Number of sentences:

The number of sentences in the speech are, 68 of President Roosevelt, 52 of President Kennedy, and 68 of President Nixon has been depicted in Figure 2.3.

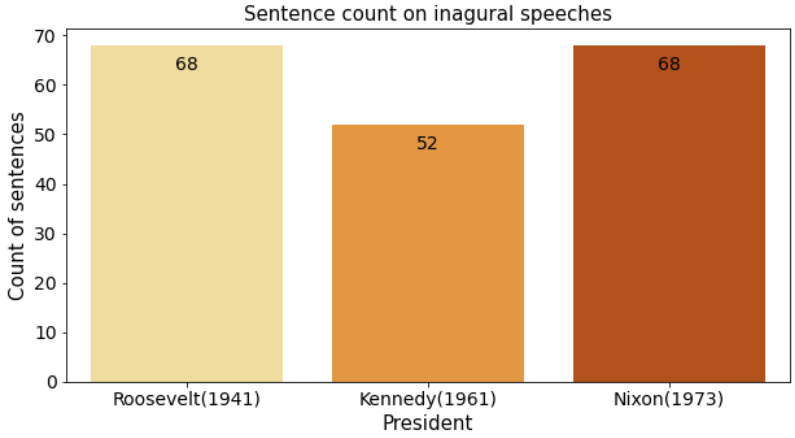


Figure 2.4: Sentence counts on inaugural speeches given by three Presidents

## 2.2 Remove all the stopwords from all three speeches.

The preprocessing of the documents include the conversion of the type case into lower, removal of all the punctuations, and the stop words. The 'nltk' package has a folder named 'corpus' which contains stop words of different languages. In the present study English language has been specifically considered the stop words.

The steps are as follows:

```
inaugural_speech['text'] = inaugural_speech['text'].apply(lambda x: " ".join(x.lower() for x in x.split()))
```

```
inaugural_speech['text'] = inaugural_speech['text'].str.replace('[^\w\s]','')

inaugural_speech['text'] = inaugural_speech['text'].apply(lambda x: " ".join(x for x in x.split() if x not in stop_words))

list(inaugural_speech[inaugural_speech['president']=="Roosevelt(1941)"].text)

list(inaugural_speech[inaugural_speech['president']=="Kennedy(1961)"].text)

list(inaugural_speech[inaugural_speech['president']=="Nixon(1973)"].text)
```

## 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)

Based on Panda's series, the frequency of top three words and their counts in the inaugural speech of Roosevelt has been shown in Figure 2.5.

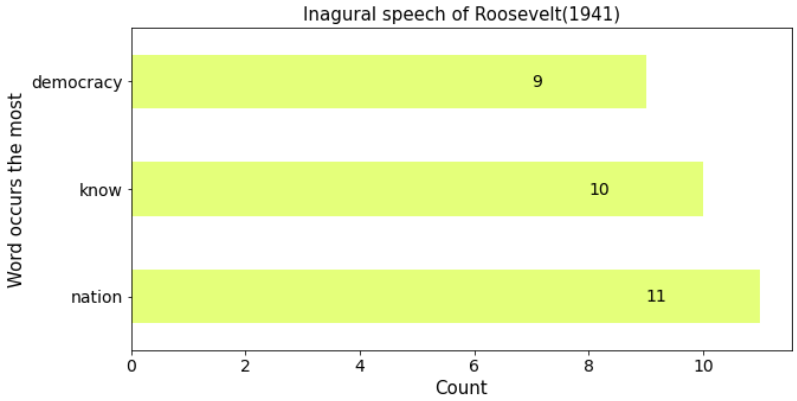


Figure 2.5: Word occurs the most number of times on inaugural speech of President Roosevelt

Based on Panda's series, the frequency of top three words and their counts in the inaugural speech of Kennedy has been shown in Figure 2.6.

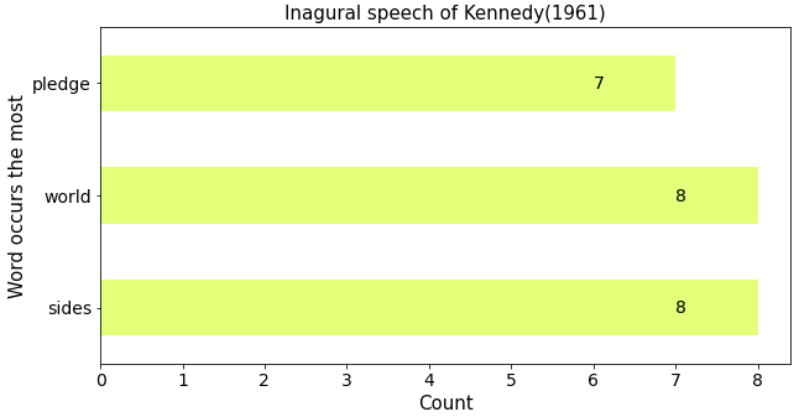


Figure 2.6: Word occurs the most number of times on inaugural speech of President Kennedy

Based on Panda's series, the frequency of top three words and their counts in the inaugural speech of Nixon has been shown in Figure 2.7.

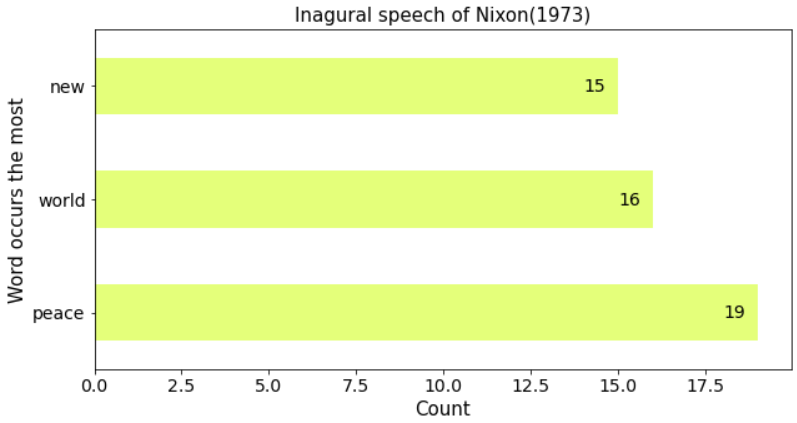


Figure 2.7: Word occurs the most number of times on inaugural speech of President Nixon

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

A word cloud (also called a tag cloud) shows how words are organized. The cloud creators highlight words based on the frequency and relevance of their use. The word cloud of each of the speeches of the variable are shown in Figure 2.8a-c.


Figure 2.8a: Word cloud of inaugural speech of President Roosevelt

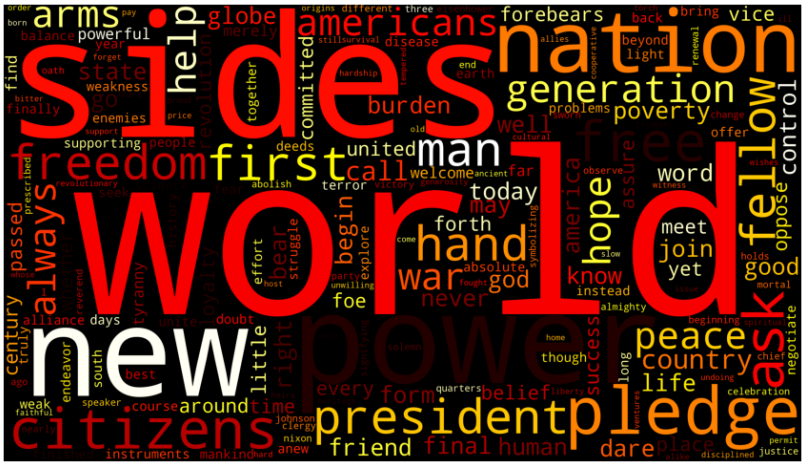


Figure 2.8b: Word cloud of inaugural speech of President Kennedy


Figure 2.8c: Word cloud of inaugural speech of President Nixon