# ML PROJECT BUSSINESS REPORT

Kratik Mehta

PGP-DSBA  Feb 2022

# Table of Contents

# List of Figures

# List of Tables

# Problem 1

## Executive Summary

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.

### Sample of the Election Dataset.

The below table shows the first 5 rows of the dataset, after removing the **Unnamed: 0** column as it only contained the row numbers.

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

*Table 1: Sample of the Election Dataset*

### Data Dictionary

1. **vote**: Party choice (Conservative or Labour)
2. **age**: Age of the respondents in years
3. **economic.cond.national**: Assessment of current national economic conditions (1 to 5).
4. **economic.cond.household**: Assessment of current household economic conditions (1 to 5).
5. **Blair**: Assessment of the Labour leader (1 to 5).
6. **Hague**: Assessment of the Conservative leader (1 to 5).
7. **Europe**: an 11-point scale that measures respondents' attitudes toward European integration. High scores represent 'Eurosceptic' sentiment (1 to 11).
8. **political.knowledge**: Knowledge of parties' positions on European integration (0 to 3).
9. **gender**: female or male.

### Checking the types of variables in the dataset.

```
<class 'pandas.core.frame.DataFrame'>
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
```

From the above output we can see that:

- There are **1525 observations** of different respondents in the data.
- There are **9 columns** in the dataset.
- Out of the *8 predictor variables*, *1 is of object type* and *7 are of integer type*.
- The variables **economic.cond.national**, **economic.cond.household**, **Blair**, **Hague**, **Europe** and **political.knowledge** are *ordinal categorical variables encoded as integers*.
- The **age** column is the only *continuous variable*.
- The target variable **vote** is of *categorical (binary) type*.
- The dataset **has no missing values**.

### Descriptive Statistics of the dataset

| | count | unique | top | freq | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **vote** | 1517 | 2 | Labour | 1057 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |
| **age** | 1517.0 | NaN | NaN | NaN | 54.241266 | 15.701741 | 24.0 | 41.0 | 53.0 | 67.0 | 93.0 |
| **economic.cond.national** | 1517.0 | NaN | NaN | NaN | 3.245221 | 0.881792 | 1.0 | 3.0 | 3.0 | 4.0 | 5.0 |
| **economic.cond.household** | 1517.0 | NaN | NaN | NaN | 3.137772 | 0.931069 | 1.0 | 3.0 | 3.0 | 4.0 | 5.0 |
| **Blair** | 1517.0 | NaN | NaN | NaN | 3.335531 | 1.174772 | 1.0 | 2.0 | 4.0 | 4.0 | 5.0 |
| **Hague** | 1517.0 | NaN | NaN | NaN | 2.749506 | 1.232479 | 1.0 | 2.0 | 2.0 | 4.0 | 5.0 |
| **Europe** | 1517.0 | NaN | NaN | NaN | 6.740277 | 3.299043 | 1.0 | 4.0 | 6.0 | 10.0 | 11.0 |
| **political.knowledge** | 1517.0 | NaN | NaN | NaN | 1.540541 | 1.084417 | 0.0 | 0.0 | 2.0 | 2.0 | 3.0 |
| **gender** | 1517 | 2 | female | 808 | NaN | NaN | NaN | NaN | NaN | NaN | NaN |

*Table 2: Descriptive Statistics of the dataset*

Looking at the values from the above 5-point summary, we see that the **target variable vote** has two classes, with *Labour class being the most frequent class*. The mean age of the respondents is 54.24 years with standard deviation of 15.7. Also, the mean and median of the **age** variable are close to each other indicating that this variable has *very low skewness*.

The **economic.cond.national**, **economic.cond.household**, **Blair** and **Hague** variables have 5 ordinal values from 1 to 5. The **Europe** variable has the highest ordinal values from 1 to 11, while the **political.knowledge** variable has 4 ordinal values from 0 to 3. The dataset has more female voters than male voters.

### Checking for Duplicate rows

```
Number of duplicate entries in the dataset: 8
```

We have 8 duplicate entries in the dataset. These entries could be of different voters who gave same response to different variables or these could be repeated entries of same voters. As we have no way to verify these duplicate records, these rows have been removed from the data for further analysis.

```
Shape of the data after removing duplicate entries: (1517, 9)
```

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

### Univariate analysis

#### vote variable

```
Class percentages in vote variable:
Labour          0.69677
Conservative    0.30323
```

Almost **70% of the respondents** in the dataset have **voted for the leader from the Labour party**.

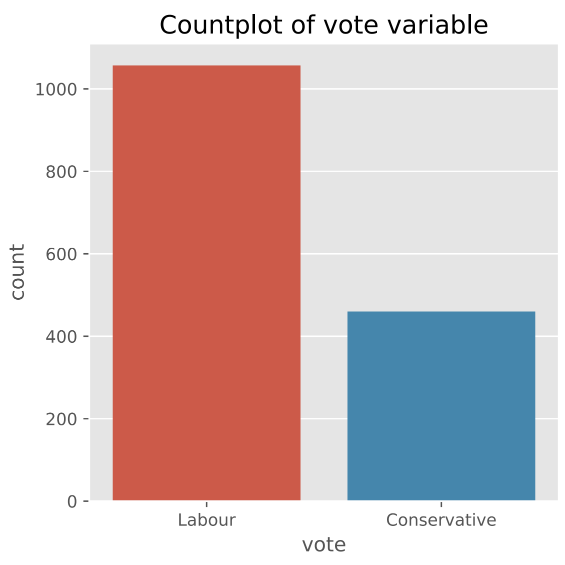


*Figure 1: Countplot of vote variable*

### age variable

```
Skewness: 0.1398
```

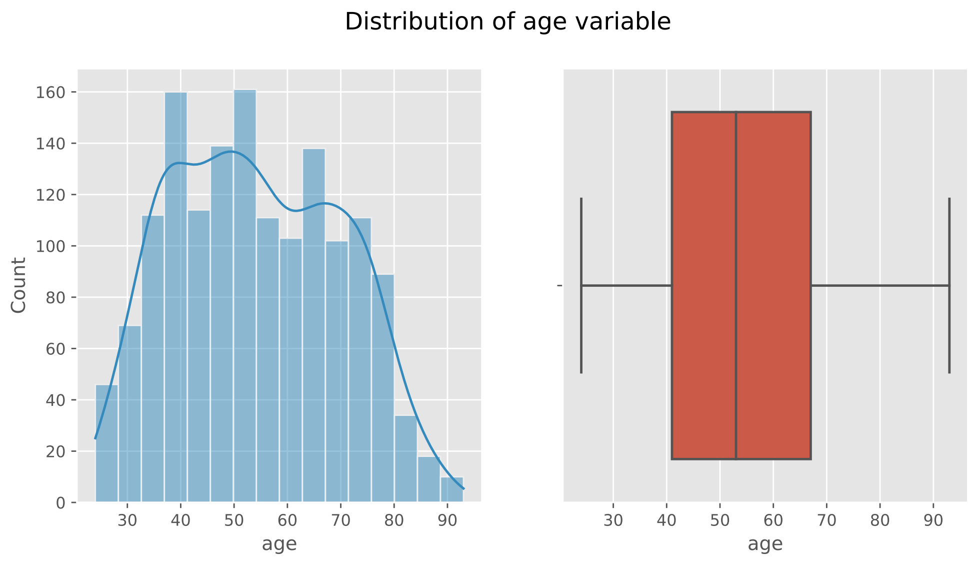


*Figure 2: Distribution of age variable*

The data in the **age** variable follows an **approximate normal distribution** with **very low positive skewness**. From the boxplot we can see that there are **no outliers** in the variable.

### economic.cond.national variable

```
Class percentages in economic.cond.national variable:
3    0.398154
4    0.354647
2    0.168754
5    0.054054
1    0.024390
```

Almost **40% of the people have given a rating of 3 followed by 35% giving a rating of 4** for the current national economic conditions.

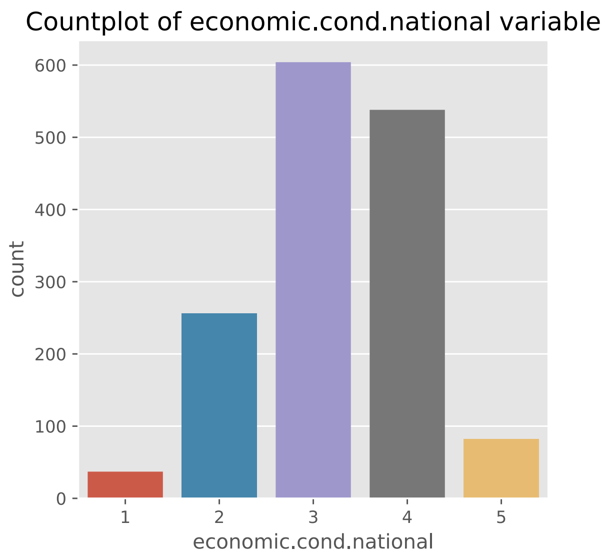


*Figure 3: Countplot of economic.cond.national variable*

### economic.cond.household variable

```
Class percentages in economic.cond.household variable:
3    0.425181
4    0.286750
2    0.184575
5    0.060646
1    0.042848
```

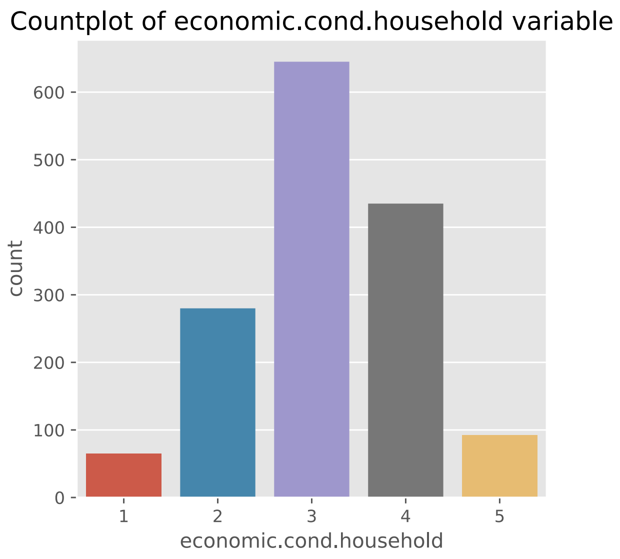


*Figure 4: Countplot of economic.cond.household variable*

Almost **43% of the people have given a rating of 3 followed by 29% giving a rating of 4** for the current household economic conditions.

### Blair variable

```
Class percentages in Blair variable:
4    0.549110
2    0.286091
5    0.100198
1    0.063942
3    0.000659
```

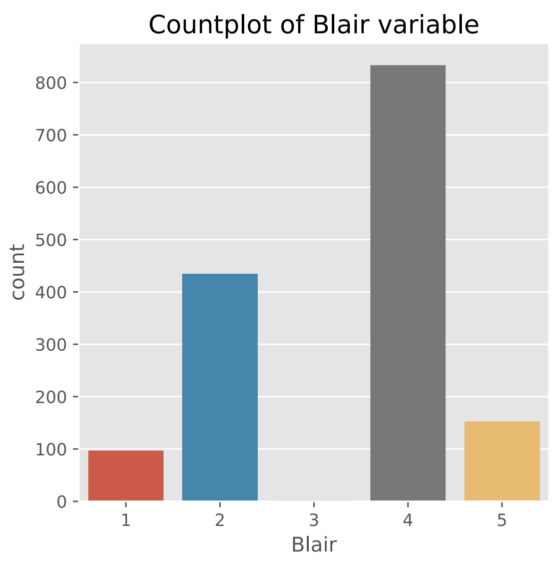


*Figure 5: Countplot of Blair variable*

Almost **55% of the voters have given a rating of 4** for the assessment of the Labour party leader i.e., **most of the voters consider Blair to be a good leader**.

## Hague variable

```
Class percentages in Hague variable:
2    0.406724
4    0.367172
1    0.153593
5    0.048121
3    0.024390
```

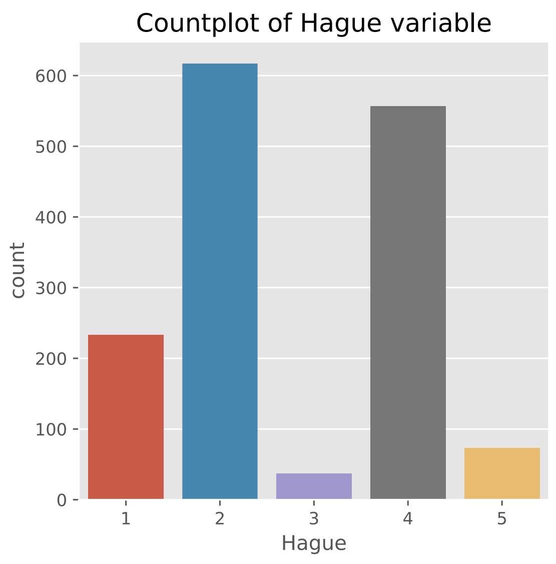


*Figure 6: Countplot of Hague variable*

About **41% of the voter have given a rating of 2 and 37% have given a rating of 4** for the assessment of the Conservative party leader i.e., **there is a mix sentiment amongst the voters regarding Hague**.

## Europe variable

```
Class percentages in Europe variable:
11    0.222808
```

```
6      0.136454
3      0.084377
4      0.083059
5      0.081081
9      0.073171
8      0.073171
1      0.071852
10     0.066579
7      0.056691
2      0.050758
```

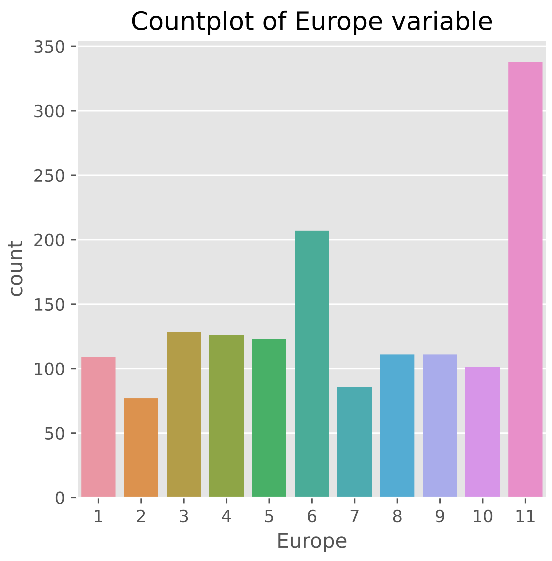


*Figure 7: Countplot of Europe variable*

Most of the people have given ratings of 11 and 6 for the European integration i.e., **most of the voters in the dataset have Eurosceptic sentiments**.

### political.knowledge variable

```
Class percentages in political.knowledge variable:
2    0.511536
0    0.299275
3    0.164140
1    0.025049
```

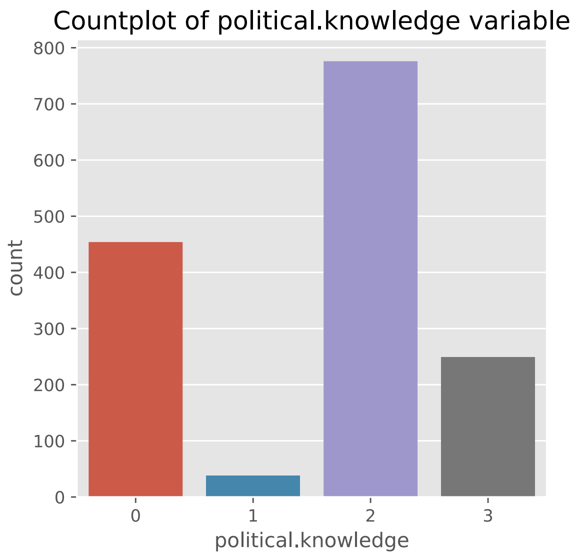


*Figure 8: Countplot of political.knowledge variable*

About **51% of the voters have good knowledge of parties' positions on European integration while 30% of the voters have no knowledge of the same**.

### gender variable

```
Class percentages in gender variable:
female     0.53263
male       0.46737
```

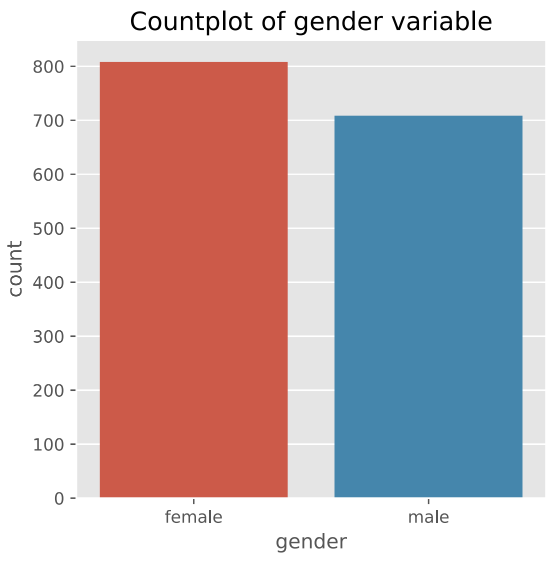


*Figure 9: Countplot of gender variable*

Approximately 53% of the voters are females i.e., **there are more females than males in the dataset**.

## Bivariate analysis

### vote vs. age

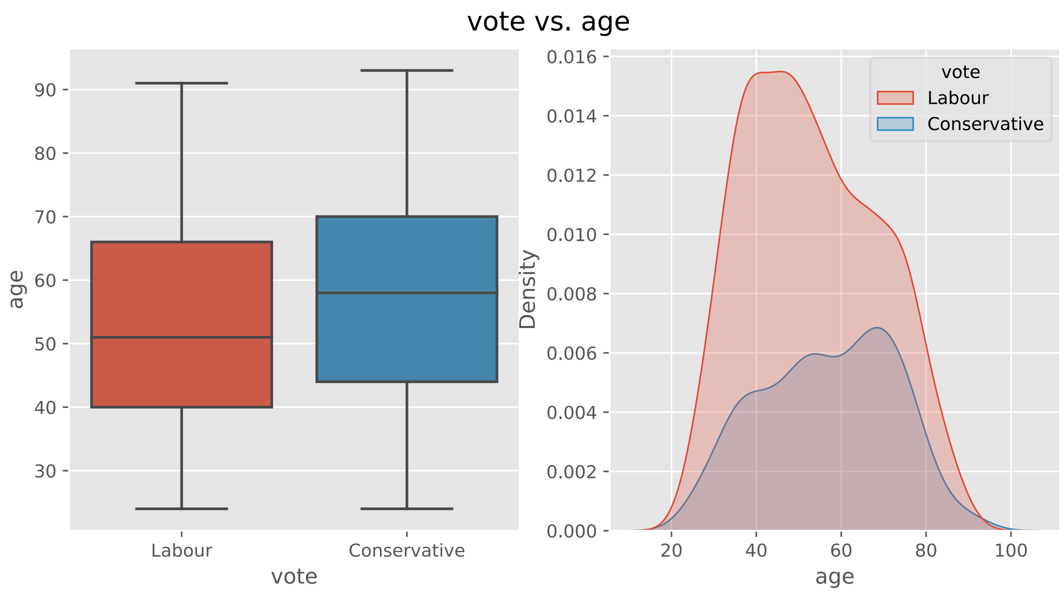


*Figure 10: vote vs. age*

The **median age of the people who voted for the Conservative party leader is greater than those who voted for the Labour party leader**. Also, the **distribution of age for the two target labels are not well separated**.

## vote vs. economic.cond.national

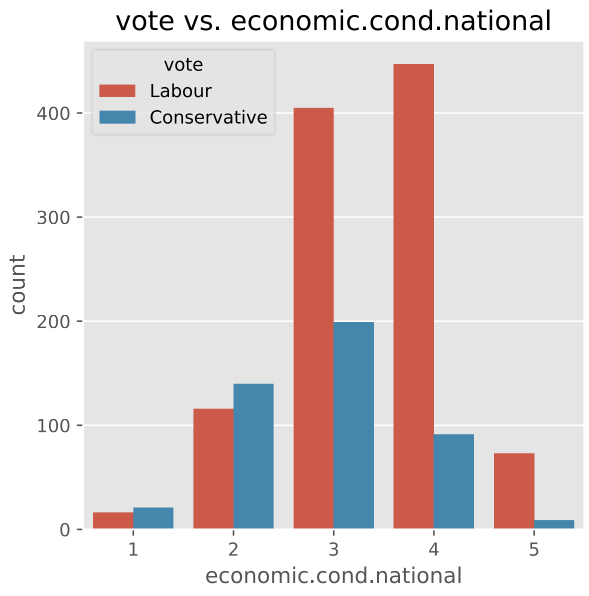


*Figure 11: vote vs. economic.cond.national*

A greater percentage of people who have given a higher rating for current national economic condition have voted for the Labour party leader while those who have given a lower rating have voted for the Conservative party leader.

## vote vs. economic.cond.household

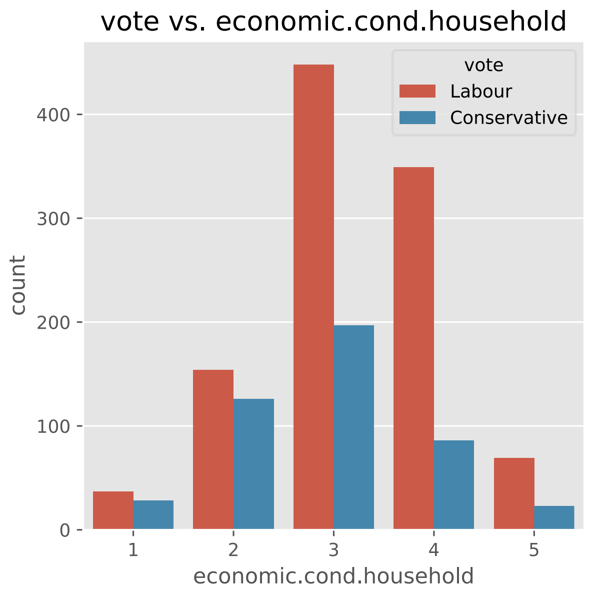


*Figure 12: vote vs. economic.cond.household*

A greater percentage of people who have given a higher rating for current household economic condition have voted for the Labour party leader while those who have given a lower rating have almost equal percentage of votes for both parties.

## vote vs. Blair

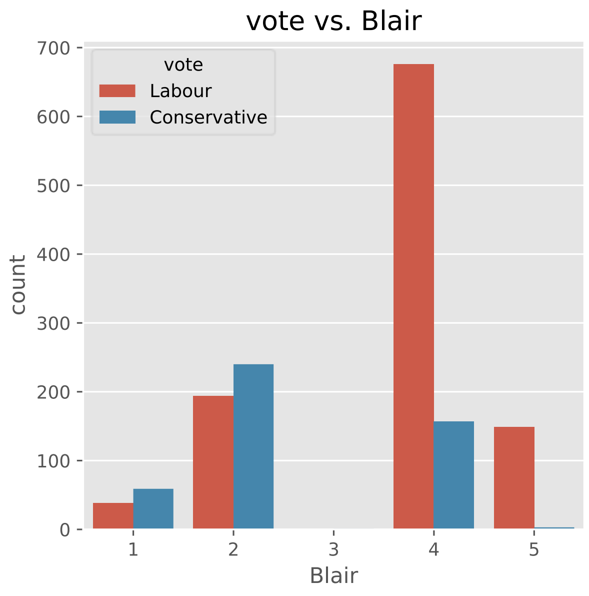


*Figure 13: vote vs. Blair*

As expected, the people who have voted for Blair have also given him higher ratings while those who have not voted for Blair have given him lower ratings.

## vote vs. Hague

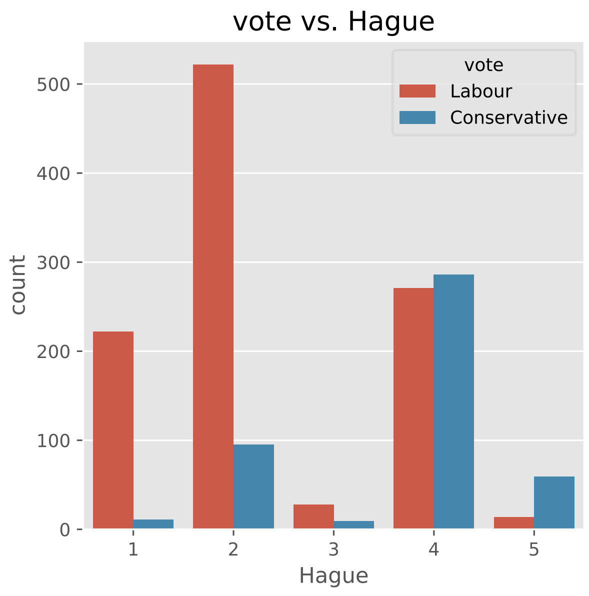


*Figure 14: vote vs. Hague*

As expected, the people who have voted for Hague have also given him higher ratings while those who have not voted for Hague have given him lower ratings.

## vote vs. Europe

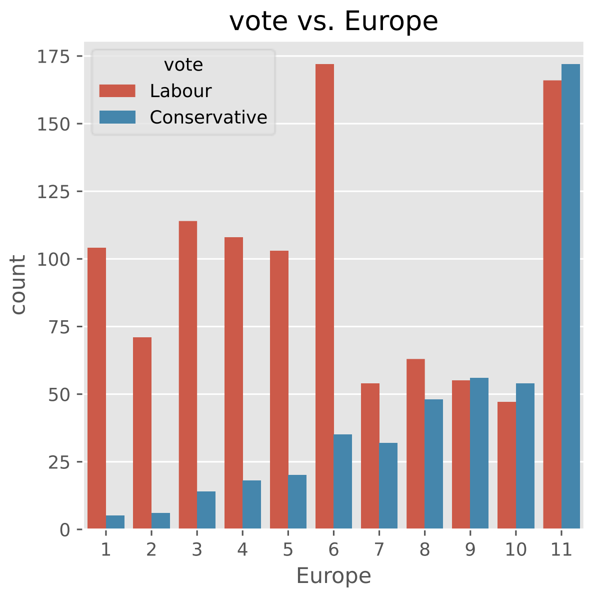


*Figure 15: vote vs. Europe*

Most of the people who have voted for Conservative party leader have given a higher rating for European integration while most of those who have voted for Labour party leader have given a lower rating.

## vote vs. political.knowledge

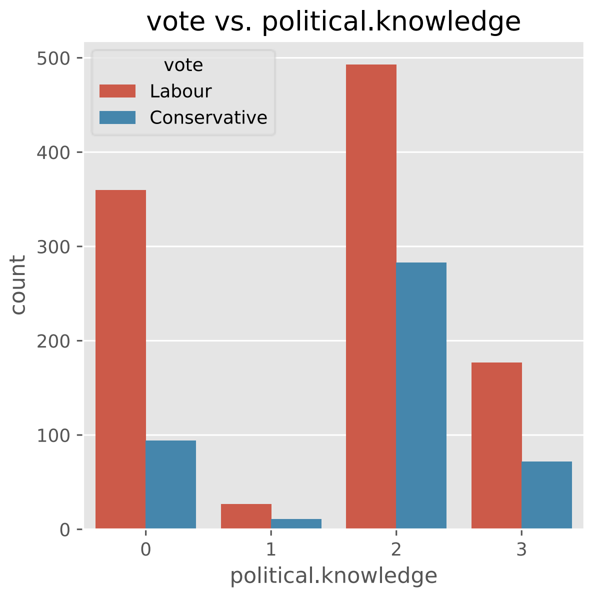


*Figure 16: vote vs. political.knowledge*

All the category ratings in the political.knowledge variable have a higher count of people who have voted for the Labour party leader.

## vote vs. gender

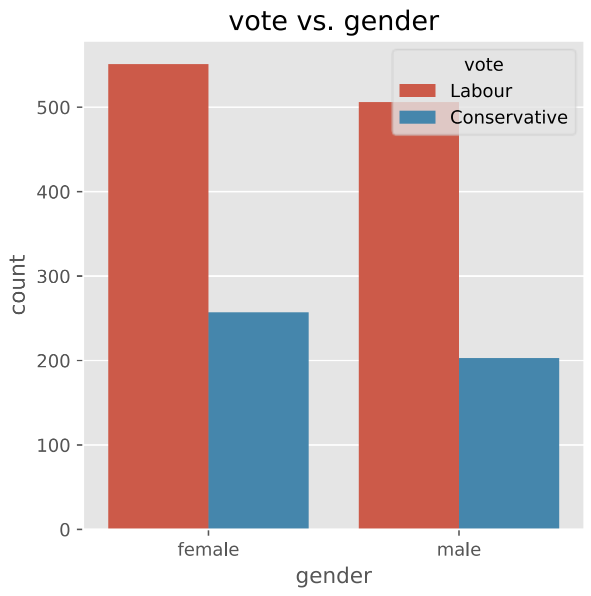
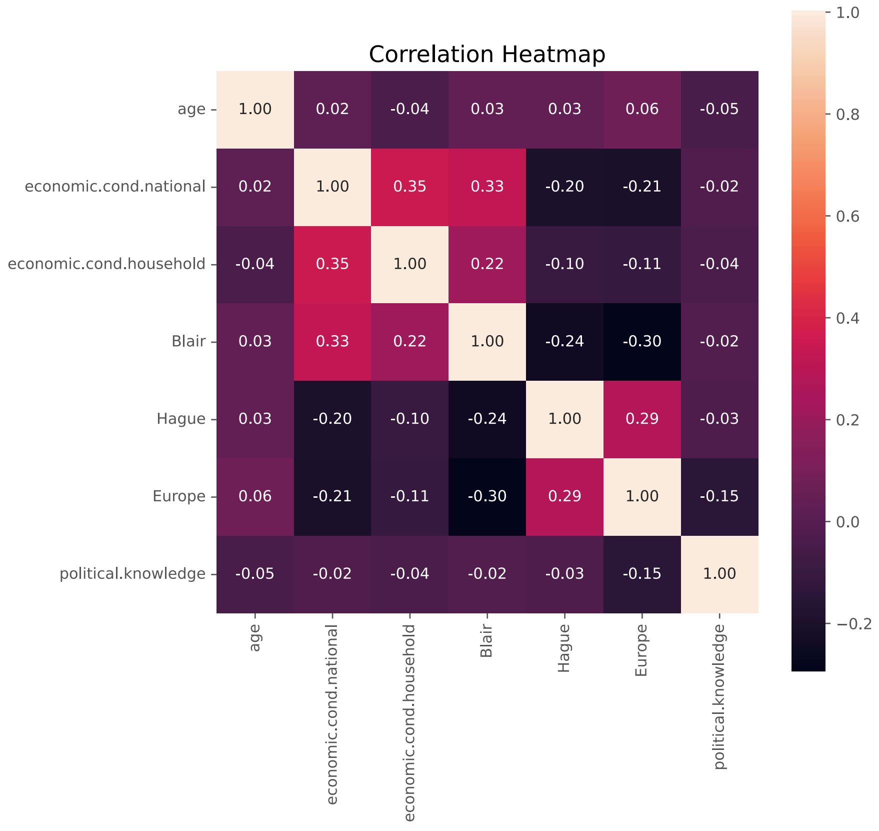


*Figure 17: vote vs. gender*

The proportion of people who have voted for Labour and Conservative party leaders in the gender variable is the same across female and male respondents.

## Correlation Heatmap



| | age | economic.cond.national | economic.cond.household | Blair | Hague | Europe | political.knowledge |
|---|---|---|---|---|---|---|---|
| age | 1.00 | 0.02 | -0.04 | 0.03 | 0.03 | 0.06 | -0.05 |
| economic.cond.national | 0.02 | 1.00 | 0.35 | 0.33 | -0.20 | -0.21 | -0.02 |
| economic.cond.household | -0.04 | 0.35 | 1.00 | 0.22 | -0.10 | -0.11 | -0.04 |
| Blair | 0.03 | 0.33 | 0.22 | 1.00 | -0.24 | -0.30 | -0.02 |
| Hague | 0.03 | -0.20 | -0.10 | -0.24 | 1.00 | 0.29 | -0.03 |
| Europe | 0.06 | -0.21 | -0.11 | -0.30 | 0.29 | 1.00 | -0.15 |
| political.knowledge | -0.05 | -0.02 | -0.04 | -0.02 | -0.03 | -0.15 | 1.00 |

*Figure 18: Correlation Heatmap*

From the above correlation plot we see that:

1. The variables economic.cond.household and economic.cond.national, Blair and economic.cond.national, and Europe and Hague have a moderate positive correlation.
2. The variables Blair and Hague, and Blair and Europe have a moderate negative correlation.
3. The other correlation values are negligible.

## Pair Plot

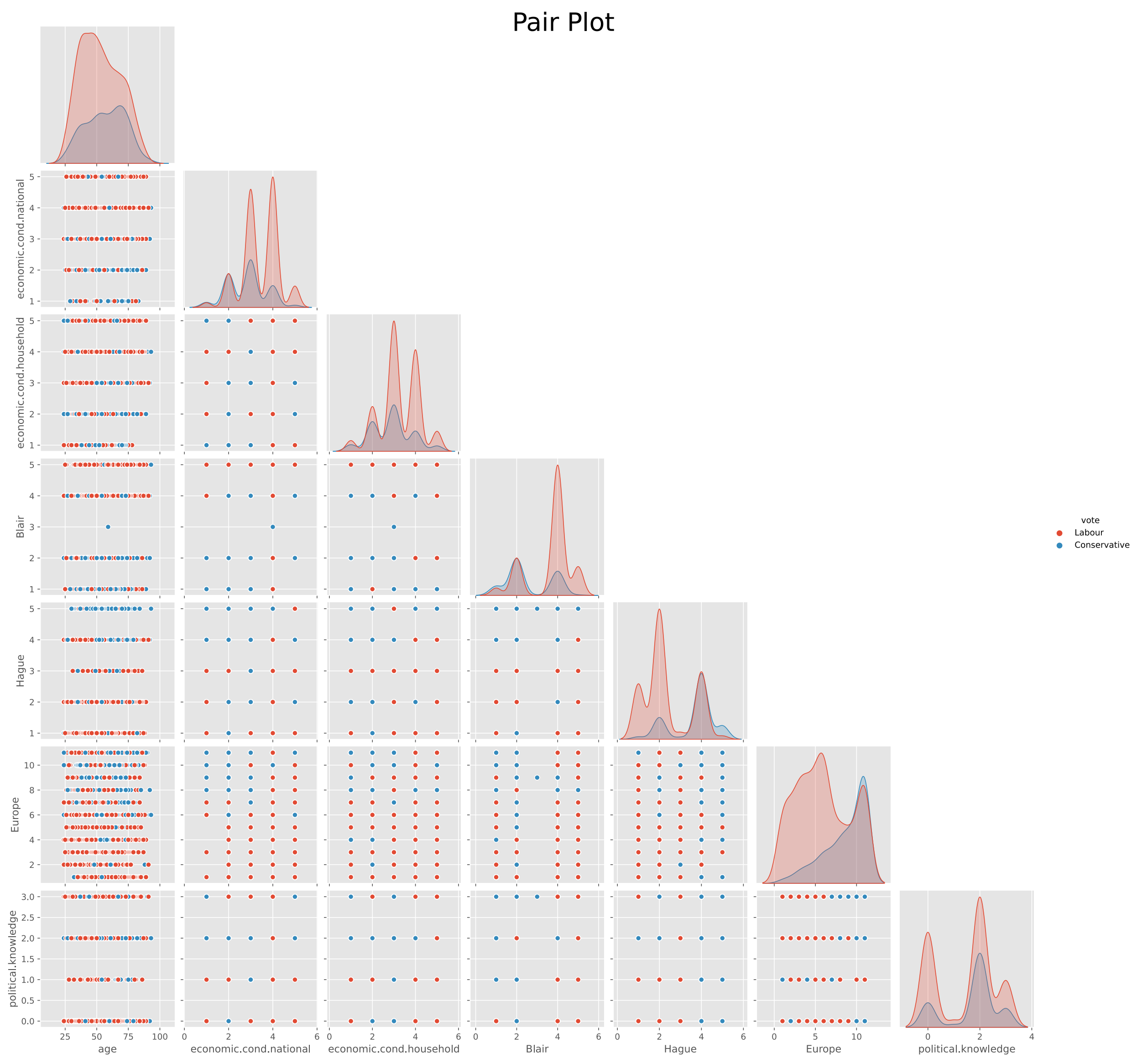


*Figure 19: Pair Plot*

The scatterplots show that there is **no multicollinearity** between the variables. Also, the kde-plot shows that **distributions of the two target classes overlap** each other. Therefore, LDA model might not perform well on this data.

## 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).

## Encoding the categorical variables

Converting the string type variables to integer variables is essential as most of the machine learning algorithms cannot handle text directly. The **economic.cond.national**, **economic.cond.household**, **Blair**, **Hague**, **Europe** and **political.knowledge** variables have already been encoded using ordinal encoding. The remaining categorical variables **vote** and **gender** are one-hot encoded as follows:

```
Encoding for vote variable: {'Labour': 0, 'Conservative': 1}
Encoding for gender variable: {'female': 0, 'male': 1}
```

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

*Table 3: Dataset after encoding*

## Scaling

| | vote | age | economic.cond.national | economic.cond.household | Blair | Hague | Europe | political.knowledge | gender |
|---|---|---|---|---|---|---|---|---|---|
| **mean** | 0.303230 | 54.241266 | 3.245221 | 3.137772 | 3.335531 | 2.749506 | 6.740277 | 1.540541 | 0.467370 |
| **std** | 0.459805 | 15.701741 | 0.881792 | 0.931069 | 1.174772 | 1.232479 | 3.299043 | 1.084417 | 0.499099 |

*Table 4: Mean and Standard deviation of the dataset*

From the above table we can see that, the **scale of the variables differs from each other**. The scale of the age column differs the most. **Scaling the data is not necessary for machine learning models like Logistic Regression, Naïve Bayes', Decision Tree, etc**. **But for models like KNN (which is a distance-based model), scaling is necessary as variables with large scales will be separated with large distances and vice versa**.

We will **scale the data using the MinMaxScaler** function. This function scales the data into a range from 0 to 1. This type of scaling preserves the distribution of the continuous variables and also the steps in the ordinal levels of the categorical variables. We will scale the data after splitting it into train and test sets.

## Splitting the data into train and test sets

The data was then split into the training and test sets. This is required because we have to make sure that the model generalizes well to unseen data. Therefore, we train the model using the training set and then evaluate it on the test set. **For small datasets, the size of the test set is taken to be 20-30% of the dataset. Here we have taken 30% of the data in the test set** as mentioned in the question. Also, the stratify argument is used to make sure that the proportions of labels in the dependent variable are the same in both the training as well as the test set.

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

| | age | economic.cond.national | economic.cond.household | Blair | Hague | Europe | political.knowledge | gender |
|---|---|---|---|---|---|---|---|---|
| **635** | 0.463768 | 0.50 | 0.50 | 0.25 | 0.75 | 0.2 | 0.666667 | 0.0 |
| **1255** | 0.681159 | 0.75 | 0.50 | 0.25 | 0.75 | 1.0 | 0.666667 | 0.0 |
| **1217** | 0.753623 | 0.50 | 0.25 | 0.75 | 1.00 | 1.0 | 0.666667 | 1.0 |
| **819** | 0.391304 | 0.50 | 0.75 | 0.75 | 0.25 | 1.0 | 0.666667 | 0.0 |
| **793** | 0.072464 | 0.75 | 0.75 | 0.75 | 0.25 | 0.9 | 0.000000 | 0.0 |

*Table 5: Scaled training set*

```
Class proportions in y_train:
0    0.696513
1    0.303487
Name: vote, dtype: float64
```

```
Class proportions in y_test:
0    0.697368
1    0.302632
Name: vote, dtype: float64
```

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

### Logistic Regression Model

Logistic Regression model uses the concept of log odds to predict the class of a data point. The model calculates the probabilities of both negative and positive classes for every data point, and then predicts the class using a threshold value for probability.

A Logistic Regression model was built with the **default parameters i.e., penalty=”l2”, solver=”lbfgs”, etc**. to get the estimate of the performance of the default model without hyperparameter tuning. The following are the train and test accuracies of the model:

```
Accuracy score on the train set: 0.845
Accuracy score on the test set: 0.809
```

From the above output we see that, both the model accuracies are high i.e., **the model is not underfitting** the data. **Training accuracy is slightly higher than the testing accuracy which means that the model is slightly overfitting the data**. We will later perform hyperparameter tuning on the model. The below output shows the intercept and coefficients learned by the model:

```
Intercept: -3.44
age: 0.019
economic.cond.national: -0.255
economic.cond.household: -0.118
Blair: -0.702
Hague: 0.873
Europe: 0.221
political.knowledge: 0.473
gender: 0.051
```

The variables **Blair** and **Hague** have the *highest impact on the target variable* while the variables **age** and **gender** have the *least impact*. Also, **the coefficients of Blair and Hague features are opposite in signs to each other which is expected**.

### LDA Model

Linear Discriminant Analysis (LDA) model finds a linear combination of features that can best discriminate between the classes in the target variable. This model can be used for classification as well as dimensionality reduction.

A LDA model was built with the **default parameters i.e., solver=”svd”, etc**. to get the estimate of the performance of the default model without hyperparameter tuning. The following are the train and test accuracies of the model:

```
Accuracy score on the train set: 0.842
Accuracy score on the test set: 0.807
```

From the above output we see that, both the model accuracies are high i.e., **the model is not underfitting** the data. **Training accuracy is slightly higher than the testing accuracy which means that the model is slightly overfitting the data**. We will later perform hyperparameter tuning on the model. The LDA model performs similar to the Logistic Regression model. The below output shows the intercept and coefficients learned by the model:

```
Intercept: -3.87
age: 0.023
economic.cond.national: -0.276
economic.cond.household: -0.119
Blair: -0.855
Hague: 1.026
Europe: 0.239
political.knowledge: 0.557
gender: 0.074
```

The variables **Blair** and **Hague** have the *highest impact on the target variable* while the variables **age** and **gender** have the *least impact*. Also, **the coefficients of Blair and Hague features are opposite in signs to each other which is expected**.

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

### KNN Model

K-Nearest Neighbours algorithm is a lazy learning algorithm. For every new point, it identifies the k-nearest points using a distance metric and then uses maximum voting to predict the class of the new point.

A KNN model was built with the **default parameters i.e., n_neighbors=5, metric="minkowski", p=2, etc**. to get the estimate of the performance of the default model without hyperparameter tuning. The **data was scaled using min-max scaling before training the model**. The following are the train and test accuracies of the model:

```
Accuracy score on the train set: 0.86
Accuracy score on the test set: 0.803
```

From the above output we see that, both the model accuracies are high i.e., **the model is not underfitting** the data. **Training accuracy is slightly higher than the testing accuracy which means that the model is slightly overfitting the data**. The KNN model has the highest training accuracy but the lowest test accuracy compared to the above two models.

### Naive Bayes' Model

Naive Bayes' algorithm uses Bayes' theorem to predict the class of a new data point given we have the prior probabilities of the classes.

A Naive Bayes' model was built with the **default parameters** to get the estimate of the performance of the default model without hyperparameter tuning. The following are the train and test accuracies of the model:

```
Accuracy score on the train set: 0.835
Accuracy score on the test set: 0.82
```

From the above output we see that, both the model accuracies are high i.e., **the model is not underfitting** the data. **Training accuracy is very close to the testing accuracy which means that the model is not overfitting the data**. The Naive Bayes' model has the highest test accuracy compared to the above three models meaning it generalizes a lot better to unseen data.

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

### Model Tuning

For all the above default models, hyperparameter tuning was done using GridSearchCV. Accuracy scoring metric and 5 folds were used for cross-validation for all the models.

## Logistic Regression Model

Best parameters found for the LogisticRegression using GridSearchCV after trial and error are:

```
{'C': 0.300, 'penalty': 'l1', 'solver': 'liblinear'}
```

Here, the **C** parameter determines the regularization strength, the **penalty** parameter means the type of regularization used, the **solver** parameter is the type of algorithm used to find the decision boundary.

```
Accuracy score on the train set: 0.844
Accuracy score on the test set: 0.807
```

The accuracy scores for the tuned model are similar to the default model created above. Also, the tuned Logistic Regression model is slightly overfitting the data.

The intercept and the coefficients of the variables are shown below:

```
Intercept: -2.096
age: 0.013
economic.cond.national: -0.313
economic.cond.household: -0.171
Blair: -0.713
Hague: 0.793
Europe: 0.195
political.knowledge: 0.395
gender: 0.000
```

The variables **Blair** and **Hague** have the *highest impact* on the target variable while the variables **age** and **gender** have the *least impact*.

## LDA Model

Best parameters found for the LinearDiscriminantAnalysis using GridSearchCV after trial and error are:

```
{'solver': 'svd'}
```

Here, the **solver** parameter is the type of algorithm used to find the decision boundary.

```
Accuracy score on the train set: 0.842
Accuracy score on the test set: 0.807
```

The accuracy scores for the tuned model are the same as the default model created above. Also, the tuned LDA model is slightly overfitting the data.

The intercept and the coefficients of the variables are shown below:

```
Intercept: -3.870
age: 0.023
economic.cond.national: -0.276
economic.cond.household: -0.119
Blair: -0.855
Hague: 1.026
Europe: 0.239
political.knowledge: 0.557
gender: 0.074
```

The variables **Blair** and **Hague** have the *highest impact* on the target variable while the variables **age** and **gender** have the *least impact*.

## KNN Model

Best parameters found for the KNN model using GridSearchCV after trial and error are:

```
{'metric': 'minkowski', 'n_neighbors': 9, 'p': 1, 'weights': 'uniform'}
```

Here, the **metric** parameter is the distance metric used to calculate the distance between points. The *minkowski* distance is a general form of Euclidean and Manhattan distance. The **n_neighbors** parameter specifies the number of nearest neighbors the algorithm uses to classify a new point. The **p** parameter is the exponent used in the minkowski distance (1 means Manhattan distance and 2 means Euclidean distance). The **weights** parameter determines the weight function used in prediction.

To cross-verify the optimum value of k, multiple KNN models were built with different values of k using the scaled data and error rate of each of the models was calculated. For all of these models, Manhattan distance metric was used as a distance measure (as found by GridSearchCV). The below plot of error rate vs. k also confirms that **for 9 nearest neighbors the KNN model has the lowest misclassification error rate**.

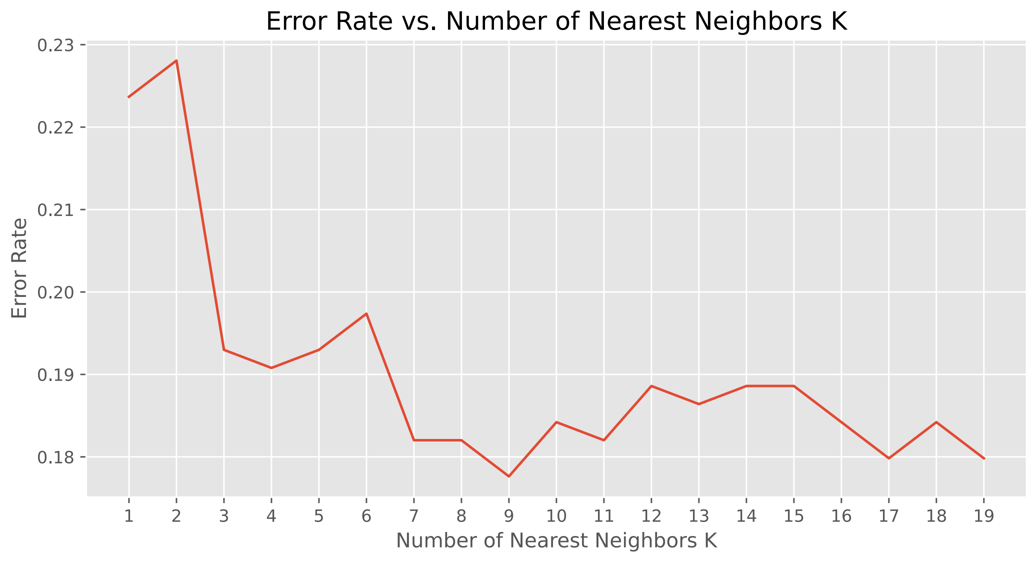


*Figure 20: Error Rate vs. Number of Nearest Neighbours K*

```
Accuracy score on the train set: 0.855
Accuracy score on the test set: 0.822
```

The accuracy scores for the tuned model are better than the default model created above. Also, overfitting in the tuned KNN model is less than the default model.

### Naive Bayes' Model

The GaussianNB estimator in the scikit-learn package has no hyperparameters that can be tuned. Hence, a Naive Bayes' model was built only using cross-validation.

```
Accuracy score on the train set: 0.835
Accuracy score on the test set: 0.82
```

The accuracy scores for the tuned model are the same as the default model created above, this is due to the fact that no hyperparameters were tuned for the model. Also, the tuned Naive Bayes' model is not overfitting the data.

## Bagging

Bagging, also known as bootstrap aggregation, is the ensemble learning method that is commonly used to reduce variance within a noisy dataset. In bagging, a random sample of data in a training set is selected with replacement. After several data samples are generated, multiple base models are then trained independently, and depending on the type of task—regression or classification, for example—the average or majority of those predictions yield a more accurate estimate. As a note, the random forest algorithm is considered an extension of the bagging method, using both bagging and feature randomness to create an uncorrelated forest of decision trees.

## Bagging Classifier

A Bagging Classifier was built on the training data using DecisionTreeClassifier as the base_estimator with max_depth of 3 to reduce overfitting. Best parameters found for the BaggingClassifier using GridSearchCV after trial and error are:

```
{'max_features': 0.800, 'max_samples': 0.700, 'n_estimators': 200}
```

Here, the **max_features** parameter determines the proportion of features used by the algorithm to build individual base models. The **max_samples** parameter determines the proportion of observations used by the algorithm to build individual base models. The **n_estimators** parameter determines the number of base estimators in the ensemble.

```
Accuracy score on the train set: 0.849
Accuracy score on the test set: 0.811
```

From the above output we see that, both the model accuracies are high i.e., **the model is not underfitting** the data. **Training accuracy is slightly higher than the testing accuracy which means that the model is slightly overfitting the data**. Overfitting was reduced by limiting the depth of individual trees to 3.

## Random Forest

Best parameters found for the RandomForestClassifier using GridSearchCV after trial and error are:

```
{'max_depth': 4, 'max_features': 0.500, 'max_samples': 1.000, 'min_samples_leaf': 4,
'min_samples_split': 30, 'n_estimators': 180}
```

Here, the **max_features** parameter determines the proportion of features used by the algorithm to build individual base models. The **max_samples** parameter determines the proportion of observations used by the algorithm to build individual base models. The **min_samples_leaf** parameter means the minimum number of samples that must be present in the leaf nodes. The **min_samples_split** parameter means the minimum number of samples that must be present in the decision nodes for splitting. The **n_estimators** parameter determines the number of trees to build in the random forest.

```
Accuracy score on the train set: 0.855
Accuracy score on the test set: 0.814
```

From the above output we see that, both the model accuracies are high i.e., **the model is not underfitting** the data. **Training accuracy is slightly higher than the testing accuracy which means that the model is slightly overfitting the data**. Random Forest model is overfitting the data slightly more than the BaggingClassifier.

| | Imp |
|---|---|
| **Hague** | 0.397144 |
| **Blair** | 0.268339 |
| **Europe** | 0.157281 |
| **political.knowledge** | 0.084654 |
| **age** | 0.040404 |
| **economic.cond.national** | 0.037823 |
| **economic.cond.household** | 0.013015 |
| **gender** | 0.001341 |

*Table 6: Feature importance for the Random Forest model*

From the above table we see that, the **Hague** variable has the *highest feature importance* followed by **Blair**, **Europe**, **political.knowledge** and **age** in *decreasing order*. The **gender** variable has the *least feature importance* in the dataset, meaning it is not a very good predictor of the dependent variable.

## Boosting

Boosting is an ensemble learning method that combines a set of weak learners into a strong learner to minimize training errors. In boosting, a random sample of data is selected, fitted with a model and then trained sequentially—that is, each model tries to compensate for the weaknesses of its predecessor. With each iteration, the weak rules from each individual classifier are combined to form one, strong prediction rule.

### AdaBoost Classifier

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. The default base estimator used by the algorithm is DecisionTreeClassifier with max_depth of 1.

Best parameters found for the AdaBoostClassifier using GridSearchCV after trial and error are:

```
{'learning_rate': 0.300, 'n_estimators': 100}
```

Here, the **learning_rate** parameter determines the weight applied to each classifier at each boosting iteration. The **n_estimators** parameter determines the number of base estimators in the ensemble.

```
Accuracy score on the train set: 0.852
Accuracy score on the test set: 0.811
```

From the above output we see that, both the model accuracies are high i.e., **the model is not underfitting** the data. **Training accuracy is slightly higher than the testing accuracy which means that the model is slightly overfitting the data**.

| | Imp |
|---|---|
| **age** | 0.30 |
| **Blair** | 0.17 |
| **Hague** | 0.16 |
| **Europe** | 0.10 |
| **economic.cond.national** | 0.09 |
| **economic.cond.household** | 0.08 |
| **political.knowledge** | 0.08 |
| **gender** | 0.02 |

*Table 7: Feature importance for the AdaBoost model*

From the above table we see that, the **age** variable has the *highest feature importance* followed by **Blair**, **Hague** and **Europe** in *decreasing order*. The **gender** variable has the *least feature importance* in the dataset, meaning it is not a very good predictor of the dependent variable.

### Gradient Boosting Classifier

In Gradient Boosting, each predictor tries to improve on its predecessor by reducing the errors. The idea behind Gradient Boosting is that instead of fitting a predictor on the data at each iteration, it actually fits a new predictor to the residual errors made by the previous predictor.

Best parameters found for the GradientBoostingClassifier using GridSearchCV after trial and error are:

```
{'learning_rate': 0.100, 'loss': 'log_loss', 'max_depth': 3, 'max_features': 'sqrt',
'min_samples_leaf': 5, 'min_samples_split': 20, 'n_estimators': 90, 'subsample': 0.500}
```

Here, the **learning_rate** parameter determines the weight applied to each classifier at each boosting iteration. The **loss** parameter determines the loss function to be optimized. The **n_estimators** parameter determines the number of base estimators in the ensemble. Remaining parameters are the same as Random Forest Classifier.

```
Accuracy score on the train set: 0.857
Accuracy score on the test set: 0.827
```

From the above output we see that, both the model accuracies are high i.e., **the model is not underfitting** the data. **Training accuracy is slightly higher than the testing accuracy which means that the model is slightly overfitting the data**.

|  | Imp |
|---|---|
| **Hague** | 0.315240 |
| **Europe** | 0.191233 |
| **Blair** | 0.174736 |
| **political.knowledge** | 0.095588 |
| **economic.cond.national** | 0.083408 |
| **age** | 0.082087 |
| **economic.cond.household** | 0.048284 |
| **gender** | 0.009423 |

*Table 8: Feature importance for the GradientBoostingClassifier model*

From the above table we see that, the **Hague** variable has the *highest feature importance* followed by **Europe**, **Blair** and **political.knowledge** in *decreasing order*. The **gender** variable has the *least feature importance* in the dataset, meaning it is not a very good predictor of the dependent variable.

## 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.

### Performance Metrics for Logistic Regression Model.

#### Accuracy Score for Training and Testing

```
Training Accuracy score for the Logistic Regression model: 0.844
Testing Accuracy score for the Logistic Regression model: 0.807
```

From the above output we see that for the **Logistic Regression model, the training and testing accuracies are close to each other**. Testing score is slightly lower than the training score, therefore **the model has slightly overfitted the data**.

#### Confusion Matrix for Training and Testing

```
Confusion matrix of the train set for the Logistic Regression model:
     0    1
0  679   60
1  106  216
```

From the above confusion matrix, we see that out of the total 322 positive examples in the training data, 216 were classified correctly while the other 106 examples were classified as negative. Here the False Negatives are greater than the False Positives, hence we can say that the **Recall is less than the Precision**.

```
Confusion matrix of the test set for the Logistic Regression model:
     0   1
0  276  42
1   46  92
```

From the above confusion matrix, we see that out of the total 138 positive examples in the testing data, 92 were classified correctly while the other 46 examples were classified as negative. Here the False Negatives are greater than the False Positives, hence we can say that the **Recall is less than the Precision**.

#### ROC AUC Score and the ROC for Training and Testing

```
ROC AUC score of train data for the Logistic Regression model: 0.894
ROC AUC score of test data for the Logistic Regression model: 0.876
```

From the above output we can see that, the AUC score for the train data is slightly more than the score for the test data; this is also an indicator of slight overfitting. An ideal classifier has the ROC curve closer to top-left corner. The below ROC curves for the Logistic Regression model are far above the diagonal line and near the top-left corner, hence it is a good classifier for the given data.

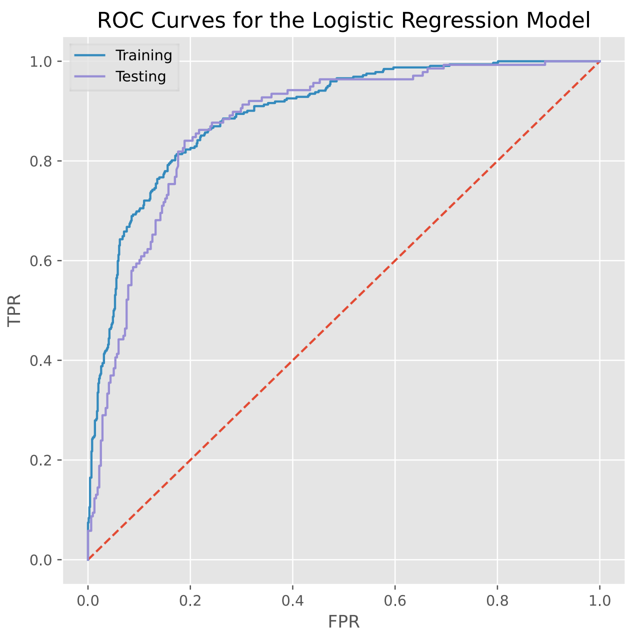


*Figure 21: ROC Curves for the Logistic Regression Model*

### Classification Reports for Training and Testing

```
Classification report of the train data for the Logistic Regression model:
              precision    recall  f1-score   support

           0       0.86      0.92      0.89       739
           1       0.78      0.67      0.72       322

    accuracy                           0.84      1061
   macro avg       0.82      0.79      0.81      1061
weighted avg       0.84      0.84      0.84      1061
```

From the above output we see that, the precision is greater than the recall for the positive class in the training data. Also, the f1-score values for both the classes are high, hence the overall performance of the model is good.

```
Classification report of the test data for the Logistic Regression model:
              precision    recall  f1-score   support

           0       0.86      0.87      0.86       318
           1       0.69      0.67      0.68       138

    accuracy                           0.81       456
   macro avg       0.77      0.77      0.77       456
weighted avg       0.81      0.81      0.81       456
```

For the test data, the accuracy, precision, recall and f1-score have decreased from the train data. Therefore, we can again conclude that the model has slightly overfit the data.

## Performance Metrics for LDA Model.

### Accuracy Score for Training and Testing

```
Training Accuracy score for the LDA model: 0.842
Testing Accuracy score for the LDA model: 0.807
```

From the above output we see that for the **LDA model, the training and testing accuracies are close to each other**. Testing score is slightly lower than the training score, therefore **the model has slightly overfitted the data**.

## Confusion Matrix for Training and Testing

```
Confusion matrix of the train set for the LDA model:
     0    1
0  671   68
1  100  222
```

From the above confusion matrix, we see that out of the total 322 positive examples in the training data, 222 were classified correctly while the other 100 examples were classified as negative. Here the False Negatives are greater than the False Positives, hence we can say that the **Recall is less than the Precision**.

```
Confusion matrix of the test set for the LDA model:
     0   1
0  273  45
1   43  95
```

From the above confusion matrix, we see that out of the total 138 positive examples in the testing data, 95 were classified correctly while the other 43 examples were classified as negative. Here the False Positives are greater than the False Negatives, hence we can say that the **Recall is greater than the Precision**.

## ROC AUC Score and the ROC for Training and Testing

```
ROC AUC score of train data for the LDA model: 0.894
ROC AUC score of test data for the LDA model: 0.873
```

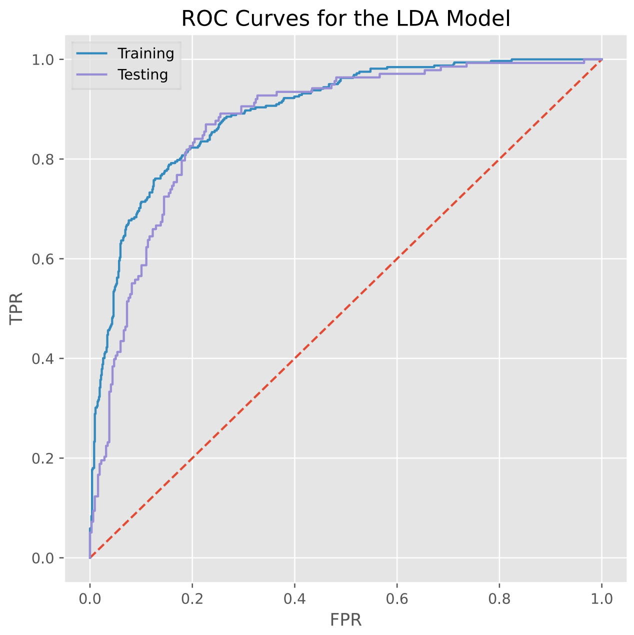


*Figure 22: ROC curves for LDA model*

From the above output we can see that, the AUC score for the train data is slightly more than the score for the test data; this is also an indicator of slight overfitting. An ideal classifier has the ROC curve closer to top-left corner. The above ROC curves for the LDA model are far above the diagonal line and near the top-left corner, hence it is a good classifier for the given data.

## Classification Reports for Training and Testing

```
Classification report of the train data for the LDA model:
```

```
              precision    recall  f1-score   support

           0       0.87      0.91      0.89       739
           1       0.77      0.69      0.73       322

    accuracy                           0.84      1061
   macro avg       0.82      0.80      0.81      1061
weighted avg       0.84      0.84      0.84      1061
```

From the above output we see that, the precision is greater than the recall for the positive class in the training data. Also, the f1-score values for both the classes are high, hence the overall performance of the model is good.

```
Classification report of the test data for the LDA model:
              precision    recall  f1-score   support

           0       0.86      0.86      0.86       318
           1       0.68      0.69      0.68       138

    accuracy                           0.81       456
   macro avg       0.77      0.77      0.77       456
weighted avg       0.81      0.81      0.81       456
```

For the test data, the accuracy, precision, recall and f1-score have decreased from the train data. Therefore, we can again conclude that the model has slightly overfit the data.

## Performance Metrics for KNN Model.

### Accuracy Score for Training and Testing

```
Training Accuracy score for the KNN model: 0.855
Testing Accuracy score for the KNN model: 0.822
```

From the above output we see that for the **KNN model, the training and testing accuracies are close to each other**. Testing score is slightly lower than the training score, therefore **the model has slightly overfitted the data**.

### Confusion Matrix for Training and Testing

```
Confusion matrix of the train set for the KNN model:
     0    1
0  679   60
1   94  228
```

From the above confusion matrix, we see that out of the total 322 positive examples in the training data, 228 were classified correctly while the other 94 examples were classified as negative. Here the False Negatives are greater than the False Positives, hence we can say that the **Recall is less than the Precision**.

```
Confusion matrix of the test set for the KNN model:
     0   1
0  279  39
1   42  96
```

From the above confusion matrix, we see that out of the total 138 positive examples in the testing data, 96 were classified correctly while the other 42 examples were classified as negative. Here the False Negatives are greater than the False Positives, hence we can say that the **Recall is less than the Precision**.

### ROC AUC Score and the ROC for Training and Testing

```
ROC AUC score of train data for the KNN model: 0.920
ROC AUC score of test data for the KNN model: 0.869
```

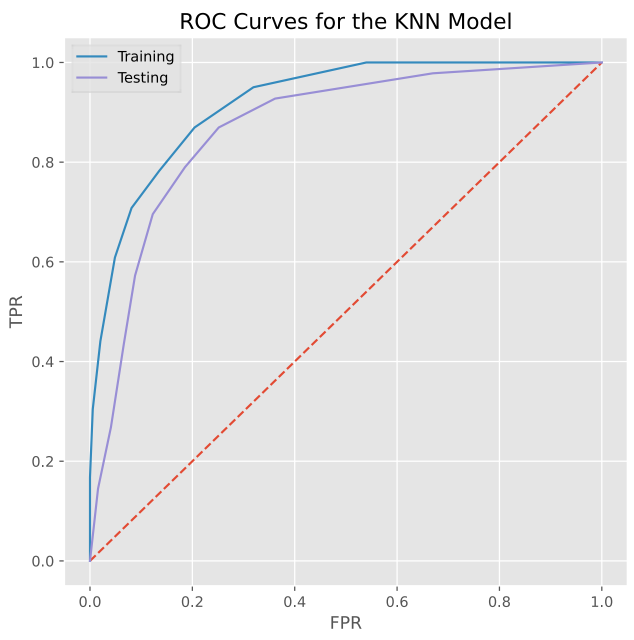


*Figure 23: ROC curves for KNN model*

From the above output we can see that, the AUC score for the train data is slightly more than the score for the test data; this is also an indicator of slight overfitting. An ideal classifier has the ROC curve closer to top-left corner. The above ROC curves for the KNN model are far above the diagonal line and near the top-left corner, hence it is a good classifier for the given data.

### Classification Reports for Training and Testing

```
Classification report of the train data for the KNN model:
              precision    recall  f1-score   support

           0       0.88      0.92      0.90       739
           1       0.79      0.71      0.75       322

    accuracy                           0.85      1061
   macro avg       0.84      0.81      0.82      1061
weighted avg       0.85      0.85      0.85      1061
```

From the above output we see that, the precision is greater than the recall for the positive class in the training data. Also, the f1-score values for both the classes are high, hence the overall performance of the model is good.

```
Classification report of the test data for the KNN model:
              precision    recall  f1-score   support

           0       0.87      0.88      0.87       318
           1       0.71      0.70      0.70       138

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

For the test data, the accuracy, precision, recall and f1-score have decreased from the train data. Therefore, we can again conclude that the model has slightly overfit the data.

## Performance Metrics for Naive Bayes' Model.

### Accuracy Score for Training and Testing

```
Training Accuracy score for the Naive Bayes' model: 0.835
```

```
Testing Accuracy score for the Naive Bayes' model: 0.820
```

From the above output we see that for the **Naive Bayes' model, the training and testing accuracies are very close to each other**. Therefore, **the model has not overfitted the data**.

## Confusion Matrix for Training and Testing

```
Confusion matrix of the train set for the Naive Bayes' model:
     0    1
0  658   81
1   94  228
```

From the above confusion matrix, we see that out of the total 322 positive examples in the training data, 228 were classified correctly while the other 94 examples were classified as negative. Here the False Negatives are greater than the False Positives, hence we can say that the **Recall is less than the Precision**.

```
Confusion matrix of the test set for the Naive Bayes' model:
     0    1
0  272   46
1   36  102
```

From the above confusion matrix, we see that out of the total 138 positive examples in the testing data, 102 were classified correctly while the other 36 examples were classified as negative. Here the False Positives are greater than the False Negatives, hence we can say that the **Recall is greater than the Precision**.

## ROC AUC Score and the ROC for Training and Testing

```
ROC AUC score of train data for the Naive Bayes' model: 0.890
ROC AUC score of test data for the Naive Bayes' model: 0.877
```

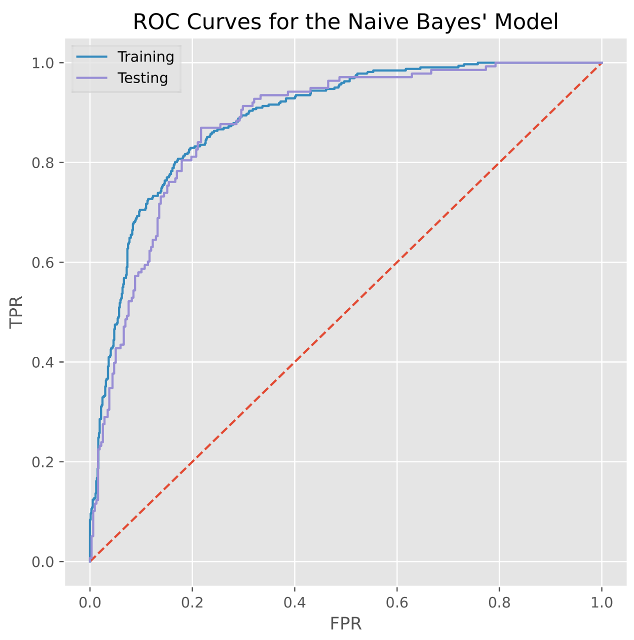


*Figure 24: ROC curves for Naive Bayes' model*

From the above output we can see that, the AUC score for the train data is very close to the score for the test data; this is also an indicator of no overfitting. An ideal classifier has the ROC curve closer to top-left corner. The above ROC curves for the Naive Bayes' model are far above the diagonal line and near the top-left corner, hence it is a good classifier for the given data.

### Classification Reports for Training and Testing

```
Classification report of the train data for the Naive Bayes' model:
              precision    recall  f1-score   support

           0       0.88      0.89      0.88       739
           1       0.74      0.71      0.72       322

    accuracy                           0.84      1061
   macro avg       0.81      0.80      0.80      1061
weighted avg       0.83      0.84      0.83      1061
```

From the above output we see that, the precision is greater than the recall for the positive class in the training data. Also, the f1-score values for both the classes are high, hence the overall performance of the model is good.

```
Classification report of the test data for the Naive Bayes' model:
              precision    recall  f1-score   support

           0       0.88      0.86      0.87       318
           1       0.69      0.74      0.71       138

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

For the test data, the accuracy, precision, recall and f1-score are very close to that of the train data. Therefore, we can again conclude that the model has not overfitted the data.

## Performance Metrics for Bagging Classifier Model.

### Accuracy Score for Training and Testing

```
Training Accuracy score for the Bagging Classifier model: 0.849
Testing Accuracy score for the Bagging Classifier model: 0.811
```

From the above output we see that for the **Bagging Classifier model, the training and testing accuracies are close to each other**. Testing score is slightly lower than the training score, therefore **the model has slightly overfitted the data**.

### Confusion Matrix for Training and Testing

```
Confusion matrix of the train set for the Bagging Classifier model:
     0    1
0  691   48
1  112  210
```

From the above confusion matrix, we see that out of the total 322 positive examples in the training data, 210 were classified correctly while the other 112 examples were classified as negative. Here the False Negatives are greater than the False Positives, hence we can say that the **Recall is less than the Precision**.

```
Confusion matrix of the test set for the Bagging Classifier model:
     0   1
0  284  34
1   52  86
```

From the above confusion matrix, we see that out of the total 138 positive examples in the testing data, 86 were classified correctly while the other 52 examples were classified as negative. Here the False Negatives are greater than the False Positives, hence we can say that the **Recall is less than the Precision**.

### ROC AUC Score and the ROC for Training and Testing

```
ROC AUC score of train data for the Bagging Classifier' model: 0.910
ROC AUC score of test data for the Bagging Classifier model: 0.880
```

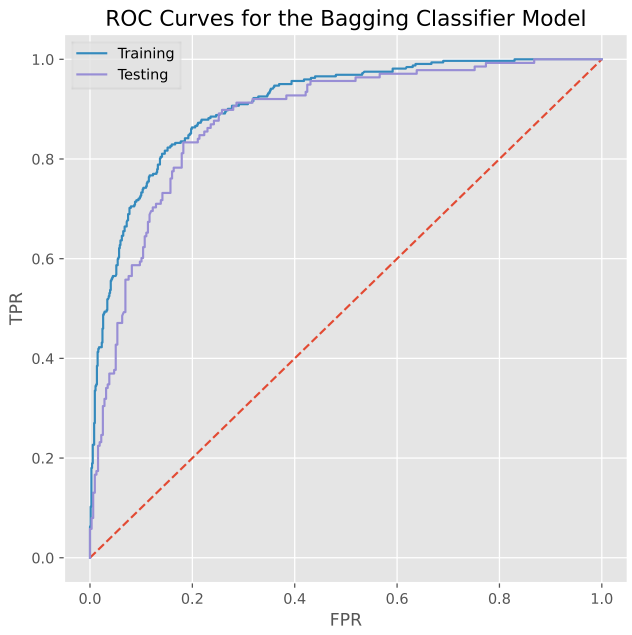


*Figure 25: ROC Curves for the Bagging Classifier Model*

From the above output we can see that, the AUC score for the train data is slightly more than the score for the test data; this is also an indicator of slight overfitting. An ideal classifier has the ROC curve closer to top-left corner. The above ROC curves for the Bagging Classifier model are far above the diagonal line and near the top-left corner, hence it is a good classifier for the given data.

## Classification Reports for Training and Testing

```
Classification report of the train data for the Bagging Classifier model:
              precision    recall  f1-score   support

           0       0.86      0.94      0.90       739
           1       0.81      0.65      0.72       322

    accuracy                           0.85      1061
   macro avg       0.84      0.79      0.81      1061
weighted avg       0.85      0.85      0.84      1061
```

From the above output we see that, the precision is greater than the recall for the positive class in the training data. Also, the f1-score values for both the classes are high, hence the overall performance of the model is good.

```
Classification report of the test data for the Bagging Classifier model:
              precision    recall  f1-score   support

           0       0.85      0.89      0.87       318
           1       0.72      0.62      0.67       138

    accuracy                           0.81       456
   macro avg       0.78      0.76      0.77       456
weighted avg       0.81      0.81      0.81       456
```

For the test data, the accuracy, precision, recall and f1-score have decreased from the train data. Therefore, we can again conclude that the model has slightly overfit the data.

## Performance Metrics for Random Forest Model.

### Accuracy Score for Training and Testing

```
Training Accuracy score for the Random Forest model: 0.855
Testing Accuracy score for the Random Forest model: 0.814
```

From the above output we see that for the **Random Forest model, the training and testing accuracies are close to each other**. Testing score is slightly lower than the training score, therefore **the model has slightly overfitted the data**.

### Confusion Matrix for Training and Testing

```
Confusion matrix of the train set for the Random Forest model:
     0    1
0  681   58
1   96  226
```

From the above confusion matrix, we see that out of the total 322 positive examples in the training data, 226 were classified correctly while the other 96 examples were classified as negative. Here the False Negatives are greater than the False Positives, hence we can say that the **Recall is less than the Precision**.

```
Confusion matrix of the test set for the Random Forest model:
     0   1
0  280  38
1   47  91
```

From the above confusion matrix, we see that out of the total 138 positive examples in the testing data, 91 were classified correctly while the other 47 examples were classified as negative. Here the False Negatives are greater than the False Positives, hence we can say that the **Recall is less than the Precision**.

### ROC AUC Score and the ROC for Training and Testing

```
ROC AUC score of train data for the Random Forest' model: 0.916
ROC AUC score of test data for the Random Forest model: 0.878
```

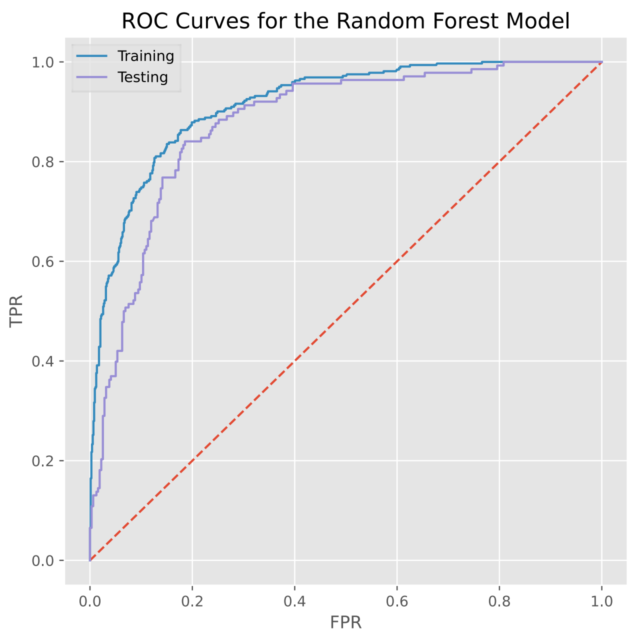


*Figure 26: ROC Curves for the Random Forest Model*

From the above output we can see that, the AUC score for the train data is slightly more than the score for the test data; this is also an indicator of slight overfitting. An ideal classifier has the ROC curve closer to top-left corner. The above ROC curves for the Random Forest model are far above the diagonal line and near the top-left corner, hence it is a good classifier for the given data.

### Classification Reports for Training and Testing

```
Classification report of the train data for the Random Forest model:
              precision    recall  f1-score   support

           0       0.88      0.92      0.90       739
           1       0.80      0.70      0.75       322

    accuracy                           0.85      1061
   macro avg       0.84      0.81      0.82      1061
weighted avg       0.85      0.85      0.85      1061
```

From the above output we see that, the precision is greater than the recall for the positive class in the training data. Also, the f1-score values for both the classes are high, hence the overall performance of the model is good.

```
Classification report of the test data for the Random Forest model:
              precision    recall  f1-score   support

           0       0.86      0.88      0.87       318
           1       0.71      0.66      0.68       138

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

For the test data, the accuracy, precision, recall and f1-score have decreased from the train data. Therefore, we can again conclude that the model has slightly overfit the data.

## Performance Metrics for AdaBoost Model.

### Accuracy Score for Training and Testing

```
Training Accuracy score for the AdaBoost model: 0.852
Testing Accuracy score for the AdaBoost model: 0.811
```

From the above output we see that for the **AdaBoost model, the training and testing accuracies are close to each other**. Testing score is slightly lower than the training score, therefore **the model has slightly overfitted the data**.

### Confusion Matrix for Training and Testing

```
Confusion matrix of the train set for the AdaBoost model:
     0    1
0  676   63
1   94  228
```

From the above confusion matrix, we see that out of the total 322 positive examples in the training data, 228 were classified correctly while the other 94 examples were classified as negative. Here the False Negatives are greater than the False Positives, hence we can say that the **Recall is less than the Precision**.

```
Confusion matrix of the test set for the AdaBoost model:
     0   1
0  278  40
1   46  92
```

From the above confusion matrix, we see that out of the total 138 positive examples in the testing data, 92 were classified correctly while the other 46 examples were classified as negative. Here the False Negatives are greater than the False Positives, hence we can say that the **Recall is less than the Precision**.

## ROC AUC Score and the ROC for Training and Testing

```
ROC AUC score of train data for the AdaBoost' model: 0.911
ROC AUC score of test data for the AdaBoost model: 0.883
```

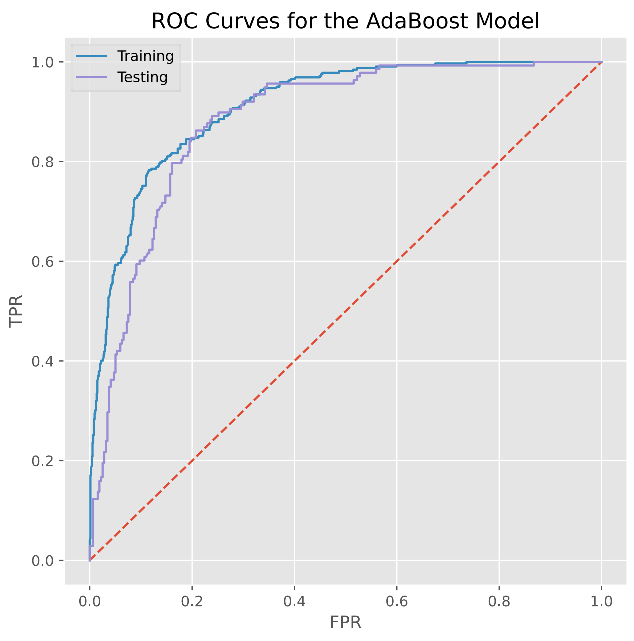


*Figure 27: ROC Curves for the AdaBoost Model*

From the above output we can see that, the AUC score for the train data is slightly more than the score for the test data; this is also an indicator of slight overfitting. An ideal classifier has the ROC curve closer to top-left corner. The above ROC curves for the AdaBoost model are far above the diagonal line and near the top-left corner, hence it is a good classifier for the given data.

## Classification Reports for Training and Testing

```
Classification report of the train data for the AdaBoost model:
              precision    recall  f1-score   support

           0       0.88      0.91      0.90       739
           1       0.78      0.71      0.74       322

    accuracy                           0.85      1061
   macro avg       0.83      0.81      0.82      1061
weighted avg       0.85      0.85      0.85      1061
```

From the above output we see that, the precision is greater than the recall for the positive class in the training data. Also, the f1-score values for both the classes are high, hence the overall performance of the model is good.

```
Classification report of the test data for the AdaBoost model:
              precision    recall  f1-score   support

           0       0.86      0.87      0.87       318
           1       0.70      0.67      0.68       138

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

For the test data, the accuracy, precision, recall and f1-score have decreased from the train data. Therefore, we can again conclude that the model has slightly overfit the data.

## Performance Metrics for Gradient Boosting Model.

### Accuracy Score for Training and Testing

```
Training Accuracy score for the Gradient Boosting model: 0.857
Testing Accuracy score for the Gradient Boosting model: 0.827
```

From the above output we see that for the **Gradient Boosting model, the training and testing accuracies are close to each other**. Testing score is slightly lower than the training score, therefore **the model has slightly overfitted the data**.

### Confusion Matrix for Training and Testing

```
Confusion matrix of the train set for the Gradient Boosting model:
     0    1
0  678   61
1   91  231
```

From the above confusion matrix, we see that out of the total 322 positive examples in the training data, 231 were classified correctly while the other 91 examples were classified as negative. Here the False Negatives are greater than the False Positives, hence we can say that the **Recall is less than the Precision**.

```
Confusion matrix of the test set for the Gradient Boosting model:
     0   1
0  278  40
1   39  99
```

From the above confusion matrix, we see that out of the total 138 positive examples in the testing data, 99 were classified correctly while the other 39 examples were classified as negative. Here the False Positives are greater than the False Negatives, hence we can say that the **Recall is greater than the Precision**.

### ROC AUC Score and the ROC for Training and Testing

```
ROC AUC score of train data for the Gradient Boosting' model: 0.923
ROC AUC score of test data for the Gradient Boosting model: 0.889
```

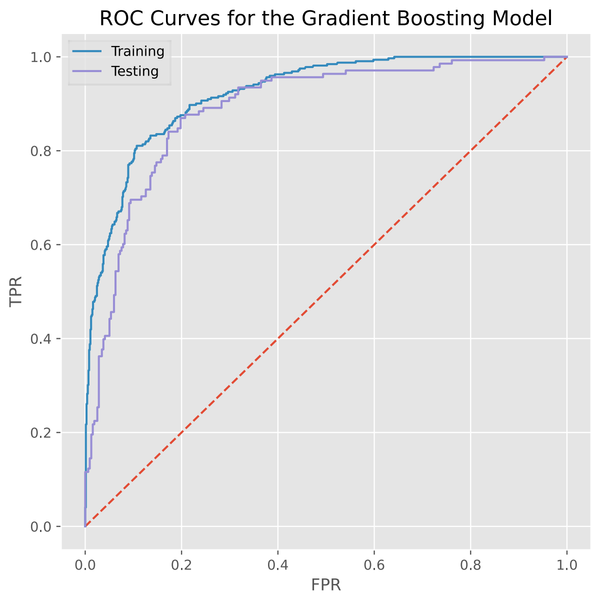


*Figure 28: ROC Curves for the Gradient Boosting Model*

From the above output we can see that, the AUC score for the train data is slightly more than the score for the test data; this is also an indicator of slight overfitting. An ideal classifier has the ROC curve closer to top-left corner. The above ROC curves for the Gradient Boosting model are far above the diagonal line and near the top-left corner, hence it is a good classifier for the given data.

### Classification Reports for Training and Testing

```
Classification report of the train data for the Gradient Boosting model:
              precision    recall  f1-score   support

           0       0.88      0.92      0.90       739
           1       0.79      0.72      0.75       322

    accuracy                           0.86      1061
   macro avg       0.84      0.82      0.83      1061
weighted avg       0.85      0.86      0.85      1061
```

From the above output we see that, the precision is greater than the recall for the positive class in the training data. Also, the f1-score values for both the classes are high, hence the overall performance of the model is good.

```
Classification report of the test data for the Gradient Boosting model:
              precision    recall  f1-score   support

           0       0.88      0.87      0.88       318
           1       0.71      0.72      0.71       138

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

For the test data, the accuracy, precision, and f1-score have decreased from the train data. Therefore, we can again conclude that the model has slightly overfit the data.

## Comparison of the models

The below two tables provide the performance metrics for all the models on the train and test data.

| | Accuracy | Precision | Recall | F1-score | AUC-score |
|---|---|---|---|---|---|
| **LR Model** | 0.844 | 0.783 | 0.671 | 0.722 | 0.894 |
| **LDA Model** | 0.842 | 0.766 | 0.689 | 0.725 | 0.894 |
| **KNN Model** | 0.855 | 0.792 | 0.708 | 0.748 | 0.920 |
| **Naive Bayes' Model** | 0.835 | 0.738 | 0.708 | 0.723 | 0.890 |
| **Bagging Classifier Model** | 0.849 | 0.814 | 0.652 | 0.724 | 0.910 |
| **RF Model** | 0.855 | 0.796 | 0.702 | 0.746 | 0.916 |
| **AdaBoost Model** | 0.852 | 0.784 | 0.708 | 0.744 | 0.911 |
| **Gradient Boosting Model** | 0.857 | 0.791 | 0.717 | 0.752 | 0.923 |

*Table 9: Performance Metrics for Training data*

| | Accuracy | Precision | Recall | F1-score | AUC-score |
|---|---|---|---|---|---|
| **LR Model** | 0.807 | 0.687 | 0.667 | 0.676 | 0.876 |
| **LDA Model** | 0.807 | 0.679 | 0.688 | 0.683 | 0.873 |
| **KNN Model** | 0.822 | 0.711 | 0.696 | 0.703 | 0.869 |
| **Naive Bayes' Model** | 0.820 | 0.689 | 0.739 | 0.713 | 0.877 |
| **Bagging Classifier Model** | 0.811 | 0.717 | 0.623 | 0.667 | 0.880 |
| **RF Model** | 0.814 | 0.705 | 0.659 | 0.682 | 0.878 |
| **AdaBoost Model** | 0.811 | 0.697 | 0.667 | 0.681 | 0.883 |
| **Gradient Boosting Model** | 0.827 | 0.712 | 0.717 | 0.715 | 0.889 |

*Table 10: Performance Metrics for Testing data*

For the given case study, predicting both the classes correctly is equally important. Therefore, f1-score is an important metric to compare on. From the above two tables we conclude that:

1. The Logistic Regression and LDA models perform very similar to each other on both train and test data.
2. The Naïve Bayes' model has least overfitted the data i.e., it generalizes well to unseen data.
3. The Gradient Boosting Classifier has the highest accuracy, f1-score and AUC score on both train and test data.

Therefore, we conclude that **the Gradient Boosting Classifier is the best model** for this case study as its overall performance is better than the other models.

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

We were tasked with building a model for a News Channel CNBE that can predict the party a voter will vote for from the given information. The dataset was read into a dataframe and the duplicate rows were removed. The variable age was the only continuous column in the dataset. Performing the EDA, we found that almost 70% of the voters have voted for the Labour party leader. From the correlation heatmap we saw that there was no multicollinearity in the dataset. The data was then split into training and test sets. Various models were built like Logistic Regression, LDA, KNN, Naïve Bayes', Bagging and Boosting. The dataset was scaled using Min Max scaling for building the KNN model. The above-mentioned models were tuned and compared on various metrics, and the best model was selected for the case study. It was found that **the Gradient Boosting Classifier is the best model** for this case study. Below are a few insights found from the data:

1. The people who have voted for the Labour party leader are of lower age group compared to those who voted for Conservative party leader. The Conservative party should also focus on younger population.
2. Most of the people who have voted for the Conservative party leader have Eurosceptic sentiment i.e., they are against the European Union. While most of the people who have voted for the Labour party leader are in favour of the European Union.
3. A large number of people who have no political knowledge have voted for the Labour party leader. These might be the voters who are really loyal the party.
4. A large number of people have given a high rating to Labour party leader Blair and a low rating to Conservative party leader Hague. This proves that the public image of a political leader is an important factor for winning an election.

# Problem 2

## Executive Summary

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

## Speeches Dataset

The three speeches were read separately and then combined into a single dataframe shown below:

| | Speech |
|---|---|
| **Roosevelt** | On each national day of inauguration since 178... |
| **Kennedy** | Vice President Johnson, Mr. Speaker, Mr. Chief... |
| **Nixon** | Mr. Vice President, Mr. Speaker, Mr. Chief Jus... |

*Table 11: Speeches Dataset*

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

The number of characters, words and sentences were calculated using different methods for all the three speeches. **The character count shown below is excluding the spaces** in the speech. **The word count includes punctuations and special characters**.

| | Speech | Word Count | Character Count | Sentence Count |
|---|---|---|---|---|
| **Roosevelt** | On each national day of inauguration since 178... | 1536 | 6249 | 68 |
| **Kennedy** | Vice President Johnson, Mr. Speaker, Mr. Chief... | 1546 | 6255 | 52 |
| **Nixon** | Mr. Vice President, Mr. Speaker, Mr. Chief Jus... | 2028 | 8223 | 69 |

*Table 12: Characters, words, and sentences counts*

In summary,

```
President Roosevelt's speech has 1536 words, 6249 characters and 68 sentences.
President Kennedy's speech has 1546 words, 6255 characters and 52 sentences.
President Nixon's speech has 2028 words, 8223 characters and 69 sentences.
```

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

### Converting the text to lowercase

The below table shows the speeches after converting them to lowercase:

| | Speech | Clean Speech |
|---|---|---|
| **Roosevelt** | On each national day of inauguration since 178... | on each national day of inauguration since 178... |
| **Kennedy** | Vice President Johnson, Mr. Speaker, Mr. Chief... | vice president johnson, mr. speaker, mr. chief... |
| **Nixon** | Mr. Vice President, Mr. Speaker, Mr. Chief Jus... | mr. vice president, mr. speaker, mr. chief jus... |

*Table 13: Speeches after converting to lower case*

### Removing Stopwords and Punctuations

The list of stopwords were taken from the nltk corpus while the list of punctuations was taken from string module**. Words like know, let and us were also included in the stopwords** as they do not have significant meanings. The speeches after removing the stopwords and punctuations are shown below:

| | Speech | Clean Speech |
|---|---|---|
| **Roosevelt** | On each national day of inauguration since 178... | national day inauguration since 1789 people re... |
| **Kennedy** | Vice President Johnson, Mr. Speaker, Mr. Chief... | vice president johnson mr speaker mr chief jus... |
| **Nixon** | Mr. Vice President, Mr. Speaker, Mr. Chief Jus... | mr vice president mr speaker mr chief justice ... |

*Table 14: Speeches after removing stopwords*

The below table shows the count of words in the speeches before and after removing the stopwords.

| | Speech | Before Word Count | After Word Count |
|---|---|---|---|
| **Roosevelt** | On each national day of inauguration since 178... | 1536 | 608 |
| **Kennedy** | Vice President Johnson, Mr. Speaker, Mr. Chief... | 1546 | 662 |
| **Nixon** | Mr. Vice President, Mr. Speaker, Mr. Chief Jus... | 2028 | 785 |

*Table 15: Word count before and after removing stopwords*

Below is a sample sentence taken from Roosevelt's original speech:

```
'On each national day of inauguration since 1789, the people have renewed their sense of
dedication to the United States.'
```

The above sentence after removing the stopwords is shown below:

```
'national day inauguration since 1789 people renewed sense dedication united states'
```

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

In President Roosevelt's speech word **nation** occurs the most number of times. Three most common words in his speech are:

```
[('nation', 11), ('spirit', 9), ('democracy', 9)]
```

In President Kennedy's speech word **world** occurs the most number of times. Three most common words in his speech are:

```
[('world', 8), ('sides', 8), ('new', 7)]
```

In President Nixon's speech word **peace** occurs the most number of times. Three most common words in his speech are:

```
[('peace', 19), ('world', 16), ('new', 15)]
```

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

In a word cloud, the size of the word depends on its frequency of occurrence. More frequent the word is, larger is its size.

### Word cloud for President Roosevelt's speech

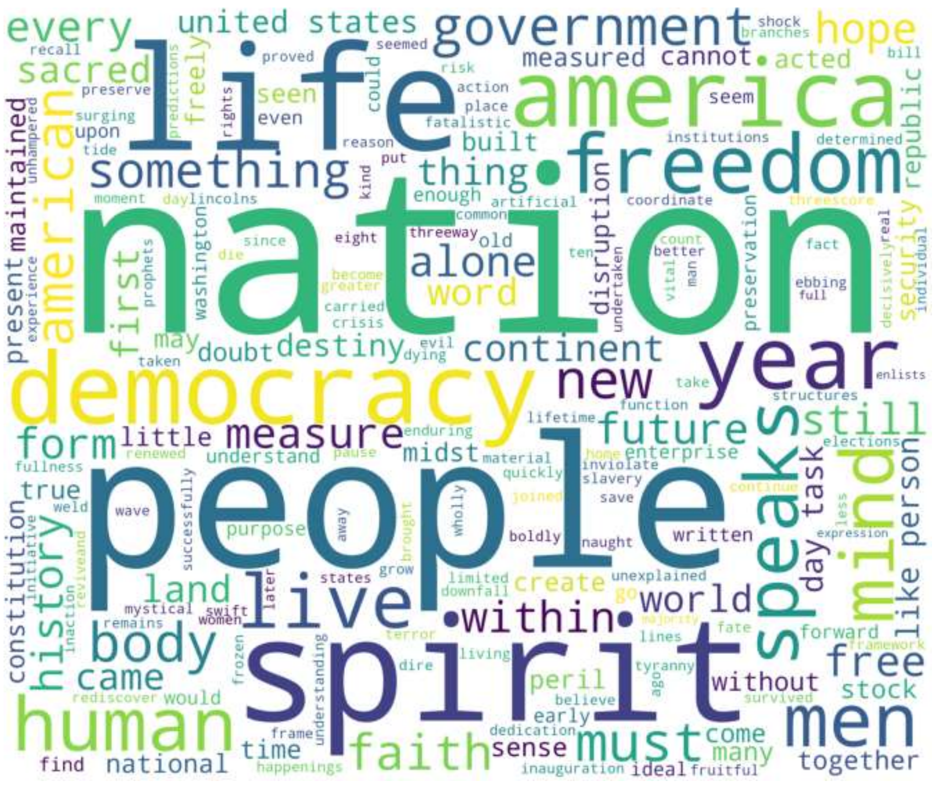


*Figure 29: Word cloud for President Roosevelt's speech*

## Word cloud for President Kennedy's speech

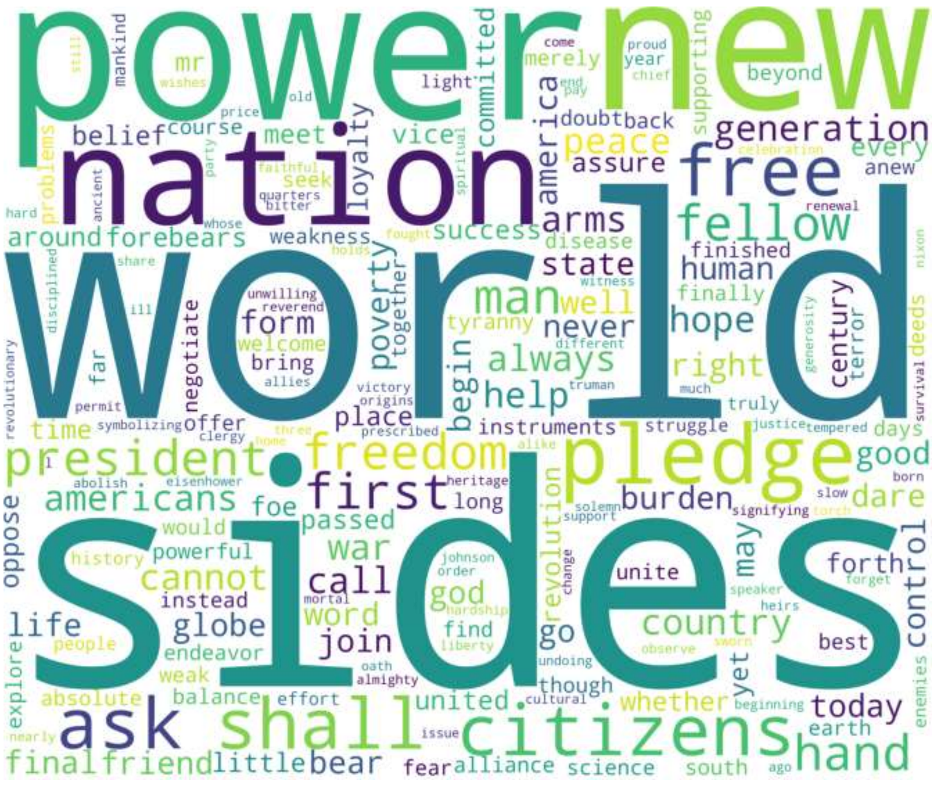


*Figure 30: Word cloud for President Kennedy's speech*

## Word cloud for President Nixon's speech

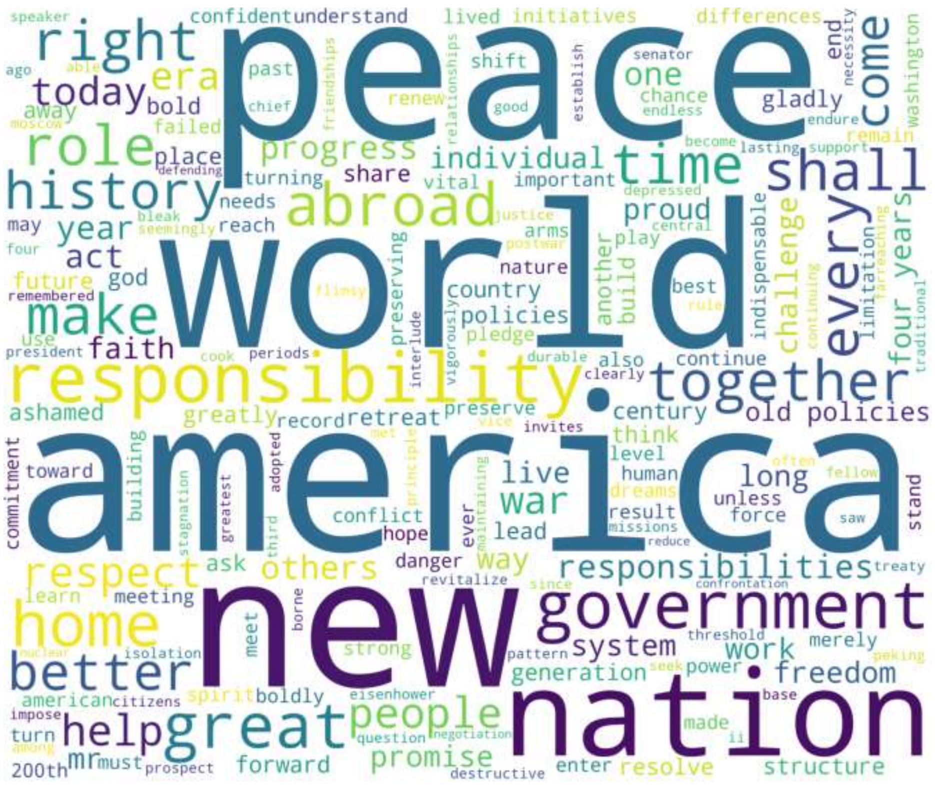


*Figure 31: Word cloud for President Nixon's speech*