greatlearning
Learning for Life

# Business Report

# DSBA

# Project - Machine Learning

**Name: Sandeep Immadi**

**Date: 03/10/2021**

greatlearning
Learning for Life

# CONTENTS:

## Problem 1:

## Problem 2:

# List of Figures:

**Problem 1:**

You are hired by one of the leading news channels CNBE who wants to analyse 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.**

Exploratory Data Analysis:

We will explore the Clustering Data set and perform the exploratory data analysis on the dataset. The major topics to be covered are below:

- Removing duplicates
- Missing value treatment
- Outlier Treatment
- Normalization and Scaling (Numerical Variables)
- Encoding Categorical variables (Dummy Variables)
- Univariate Analysis
- Bivariate Analysis

Data Insights of top 5 records after dropping unnamed:

| | 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 |

Fig.1

Data Insights of bottom 5 records:

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

Fig.2

Data Info:

```
<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
```

Fig.3

Data Summary:

| | 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 |

Fig.4

Total columns:

```
Index(['vote', 'age', 'economic.cond.national', 'economic.cond.household',
       'Blair', 'Hague', 'Europe', 'political.knowledge', 'gender'],
      dtype='object')
```

Fig.5

Data Types:

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

Fig.6

Missing Values: There are no null/missing values present in the dataset

Data Dimensions: (1525,9)

Missing values: 0

Duplicates in the data: 8 duplicates were dropped

Skewness:

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

Fig.8

Kurtosis:

```
age                       -0.943708
economic.cond.national    -0.256575
economic.cond.household   -0.209035
Blair                     -1.060248
Hague                     -1.395161
Europe                    -1.236843
political.knowledge       -1.222260
dtype: float64
```

Fig.9

Outliers :

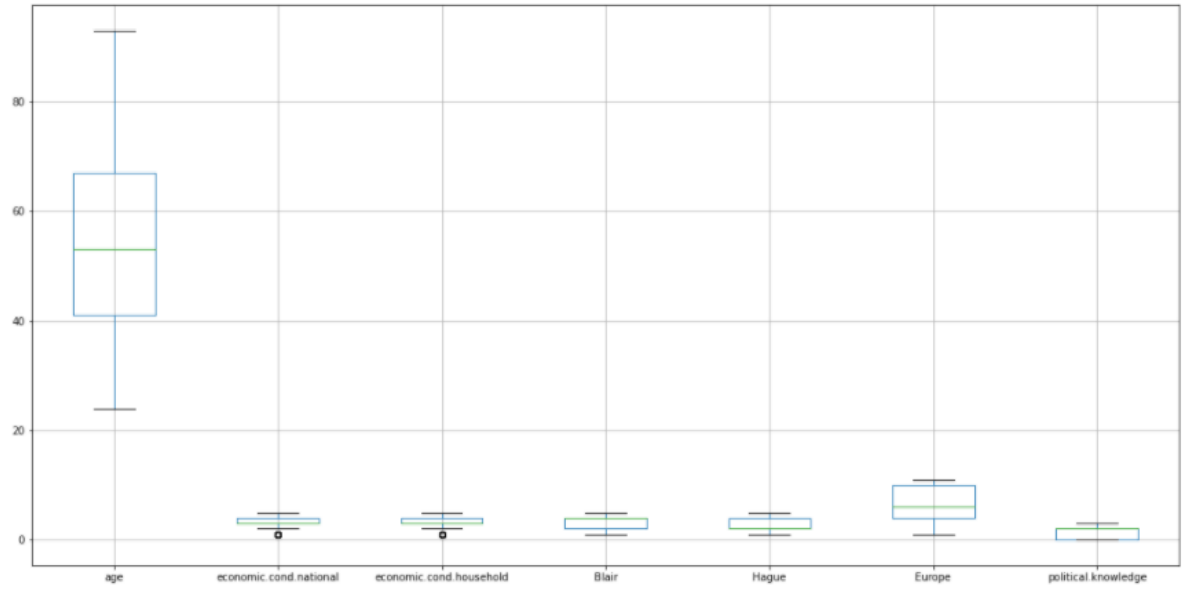


Fig.10

Histograms:

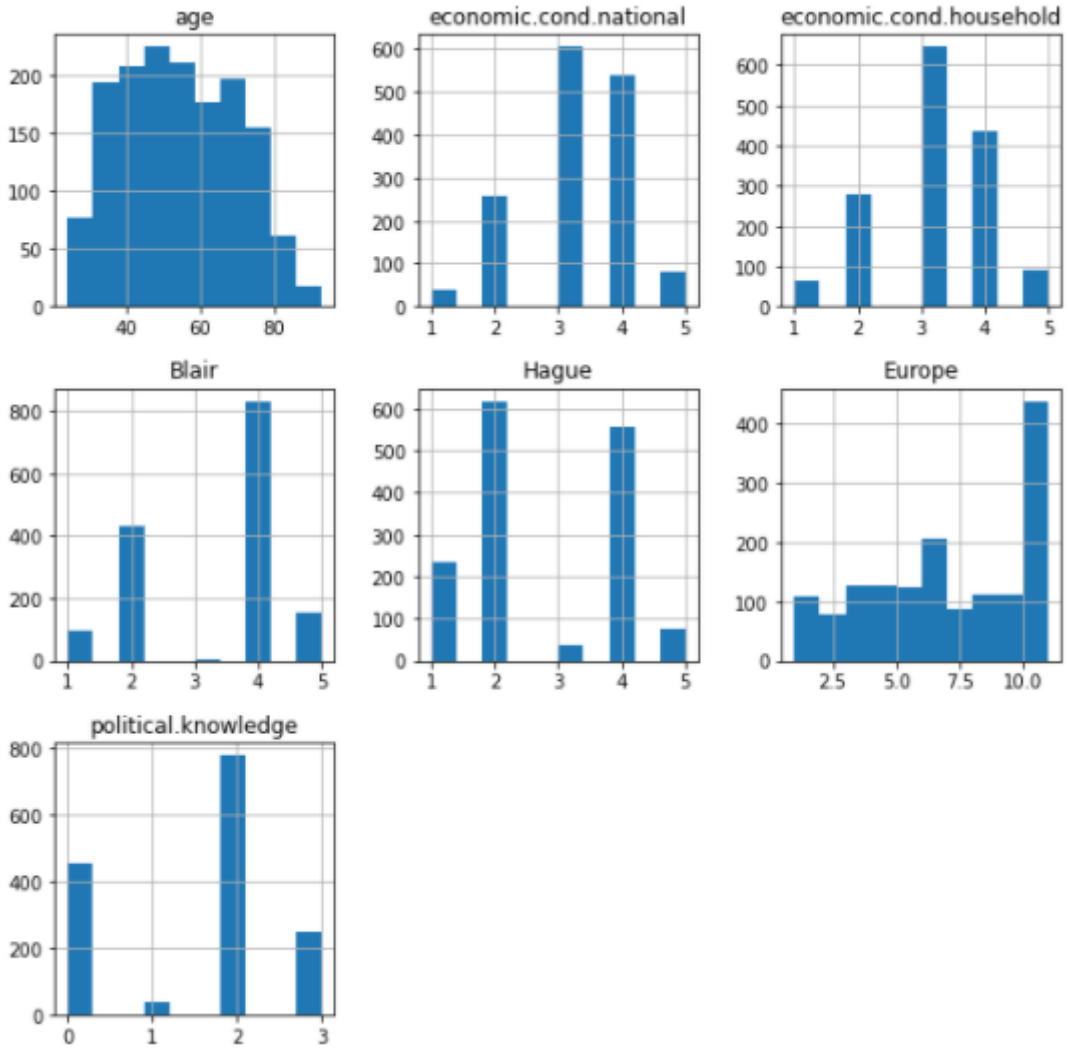


Fig.11

Inferences:

- The dataset has 9 variables; 2 categorical variables (object) and 7 continuous variables (int64).
- The variables in the data set are not all at scale.
- We can see that in the target variable, vote, the variables have following percentage of values:
- Labour: 69.71%
- Conservative: 30.29%
- There are no null values in the data set.
- There are 8 duplicate rows. Even though they could represent different person with exact same profile and political outlook, we drop these rows as they are few in number and add no value to the data set.

- The boxplot distribution of the continuous variables shows that there are marginal outliers in two variables: economic.cond.national and economic.cond.household.
- We decide not to treat these outliers as they are present in only two variables and in very small numbers.
- when looking at the values in the dataset for the other variables, they all look like categorical columns except age.
- Removed the unwanted variable “Unnamed: 0”, which is not giving a meaningful information. And displaying the head of the Election dataset.
- Not treating the outliers. outliers are to be treated for only continuous columns analyses.
- Dataset does not have null values
- The mean and median for the only integer column ‘age’ is almost same indicating the column is normally distributed.
- ‘vote’ has two unique values Labour and Conservative, which is also a dependent variable.
- ‘gender’ has two unique values male and female.
- Rest all the columns has object variables with ‘Europe’ being highest having 11 unique values.
- When we check for null values in the given dataset using the .info () and .is null () function, we get the above two tables, which clearly shows that there are no null values present.
- However, upon further checking the five-point summary, we notice that column “Political knowledge” has the minimum value as ‘0’ which may mean that there are either null values present in this column or the value will take a zero value.
- We must deep dive into the data to understand this variable further.
- Variable ‘Political knowledge’ has ordinal values where 0 represents no political knowledge and 3 represents highest level of political knowledge.
- Hence, in this case, we cannot say that there are missing values present in this variable.
- Some of the values will be taken as ‘0’ and we will consider it as numeric value.

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

**Univariate and Bivariate Analysis:**

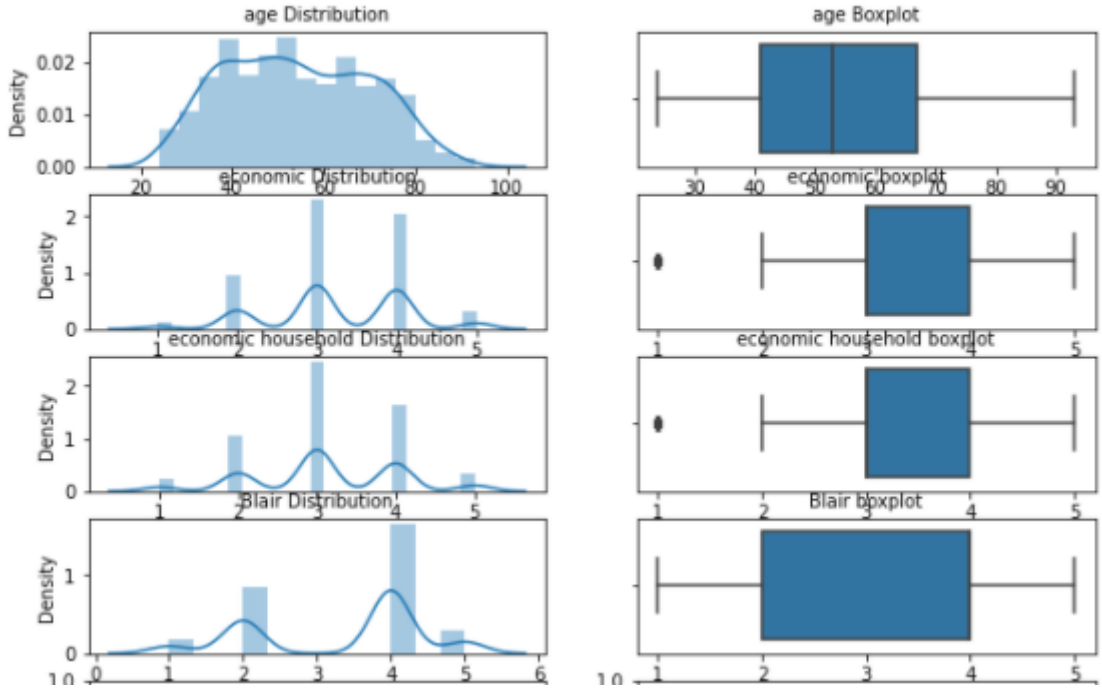


Fig.12

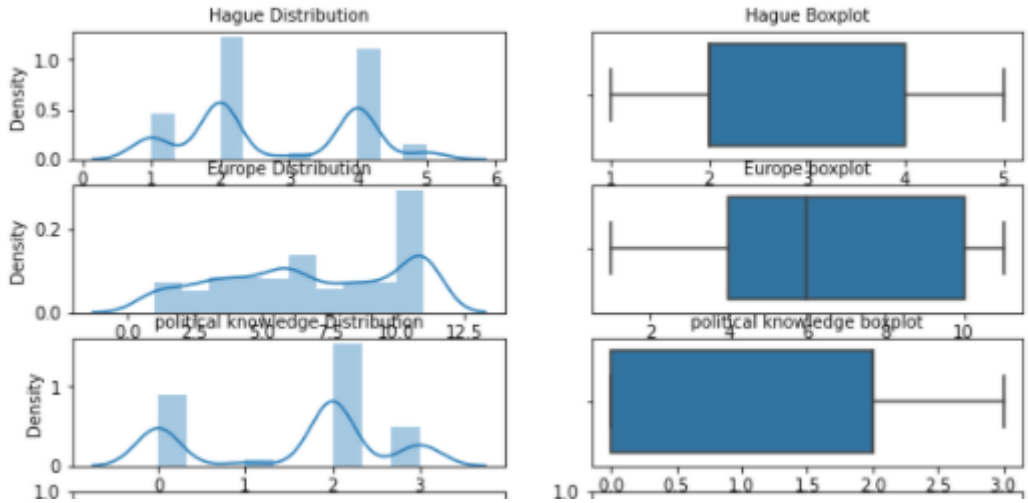


Fig.13

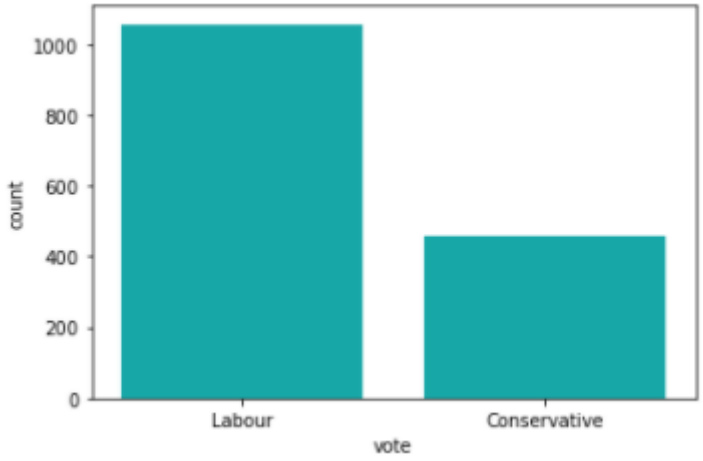


Fig.14

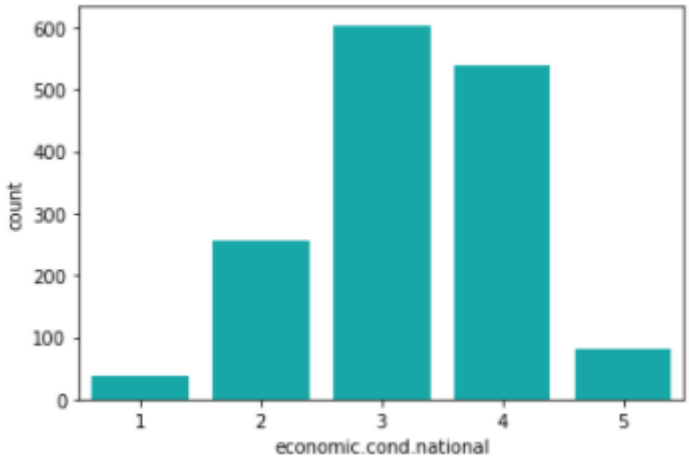


Fig.15

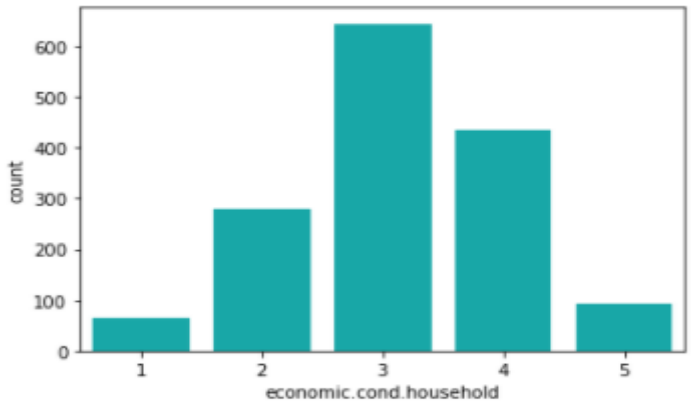


Fig.16

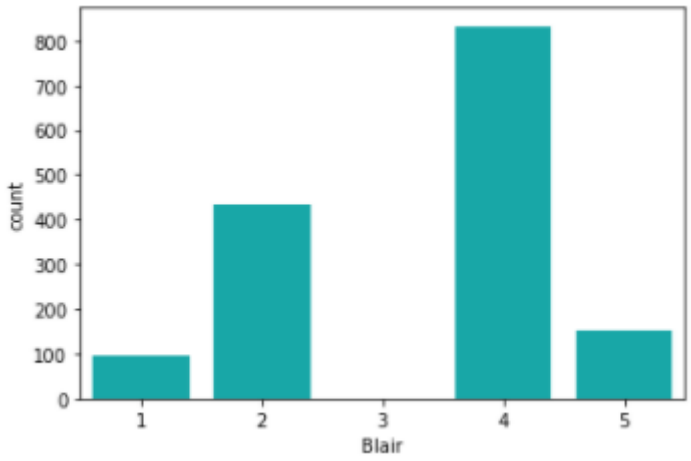


Fig.17

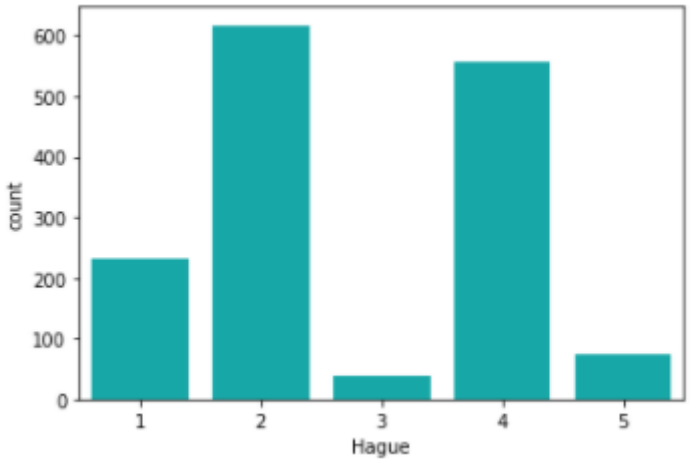


Fig.18

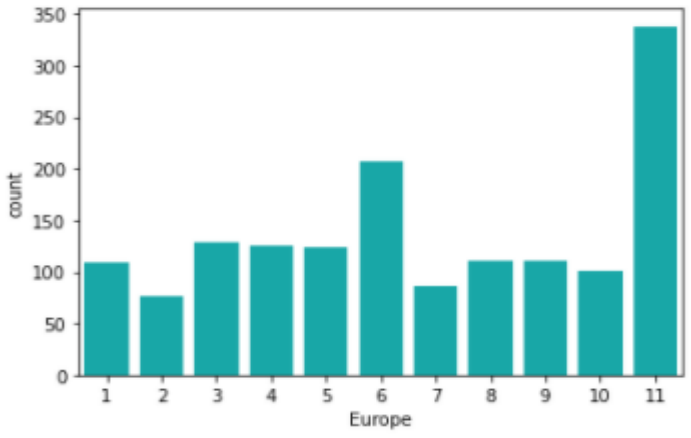


Fig.19

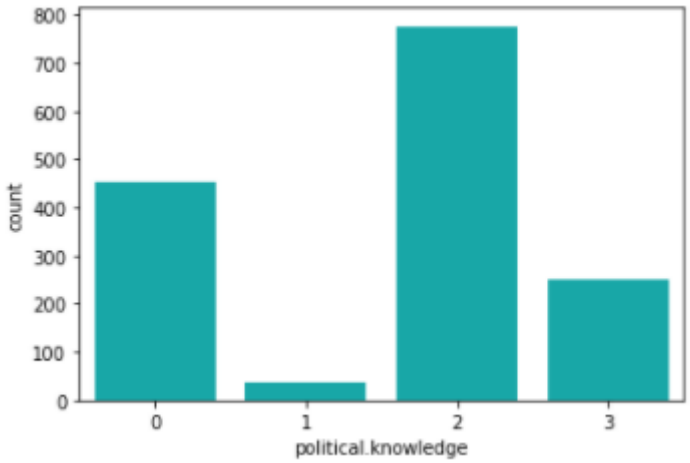
800
700
600
500
400
300
200
100
0
count
0
1
2
3
political.knowledge

Fig 20

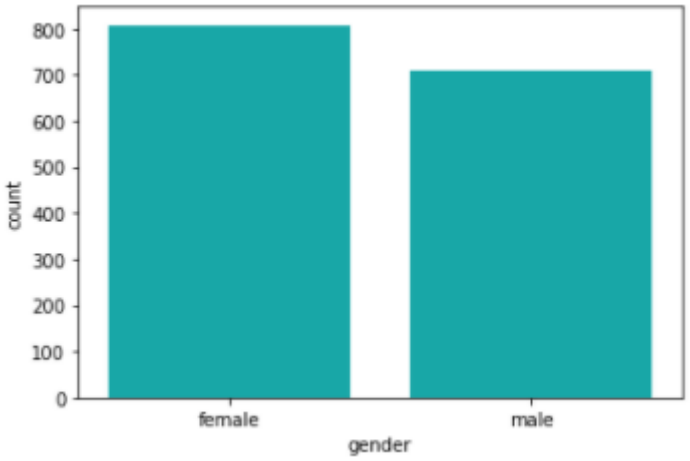
800
700
600
500
400
300
200
100
0
count
female
male
gender

Fig.21

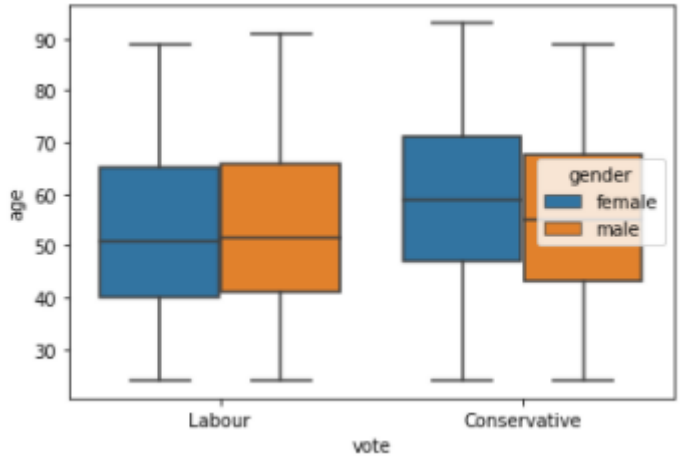


Fig.22

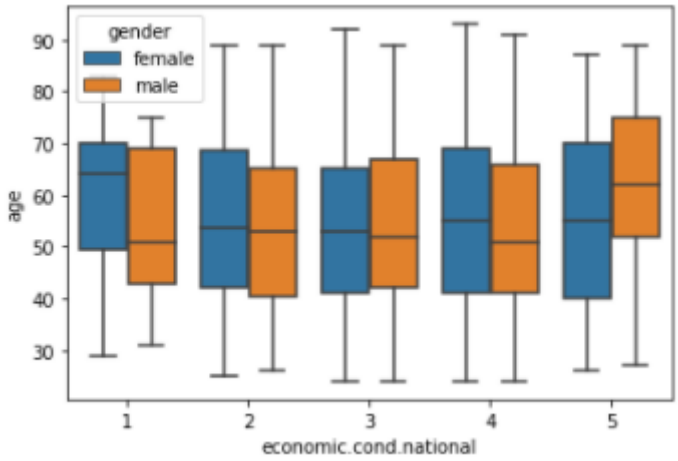


Fig.23

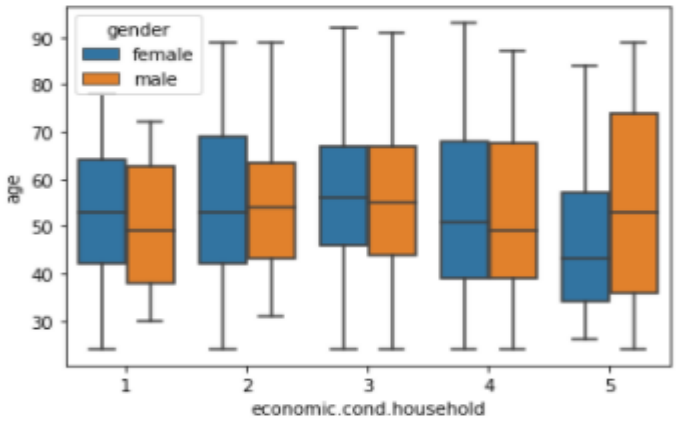


Fig.24

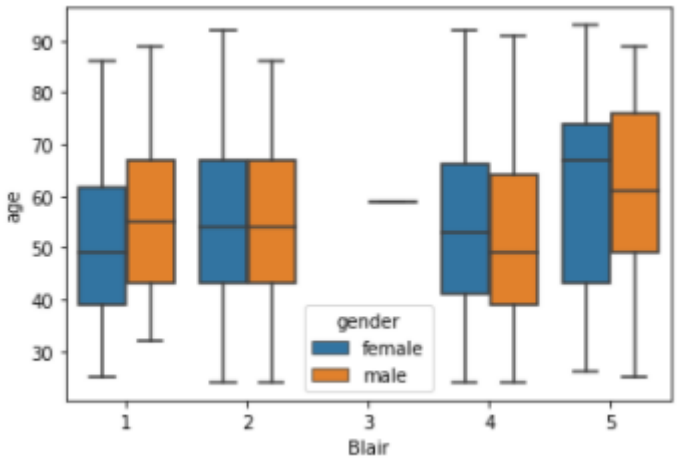


Fig.25

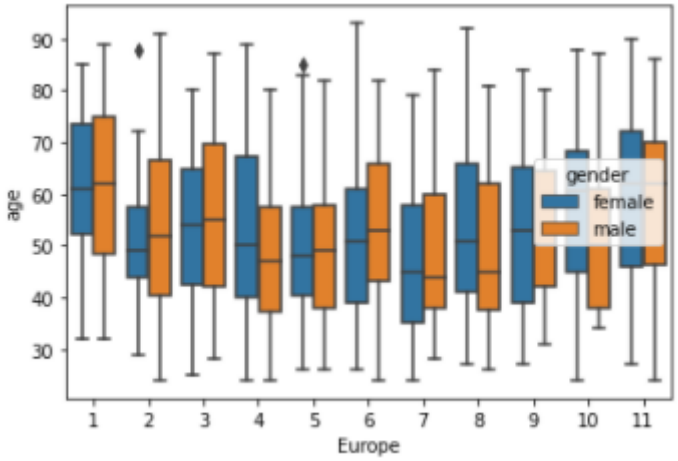


Fig.26

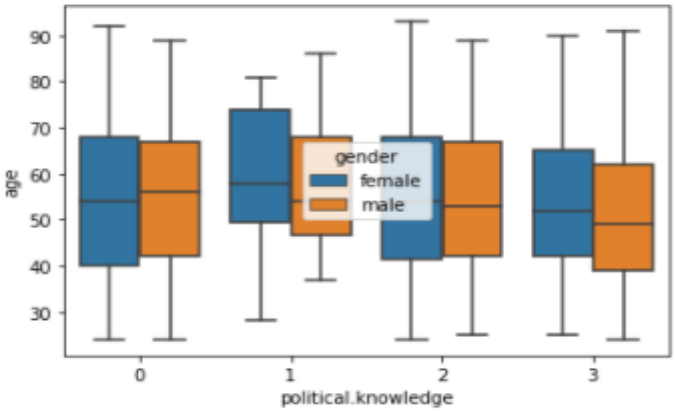


Fig.27

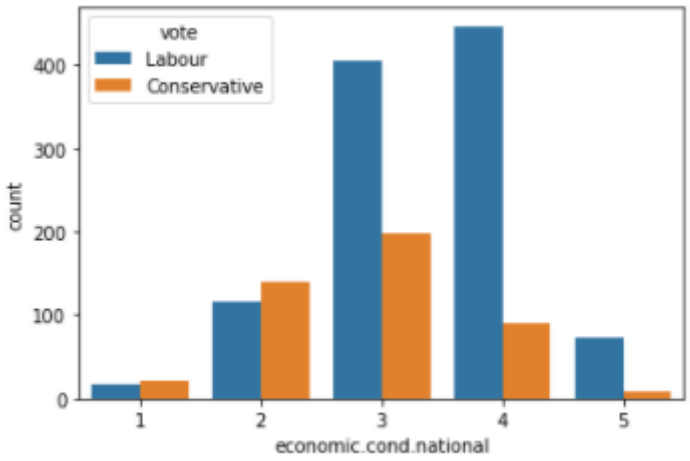


Fig.28

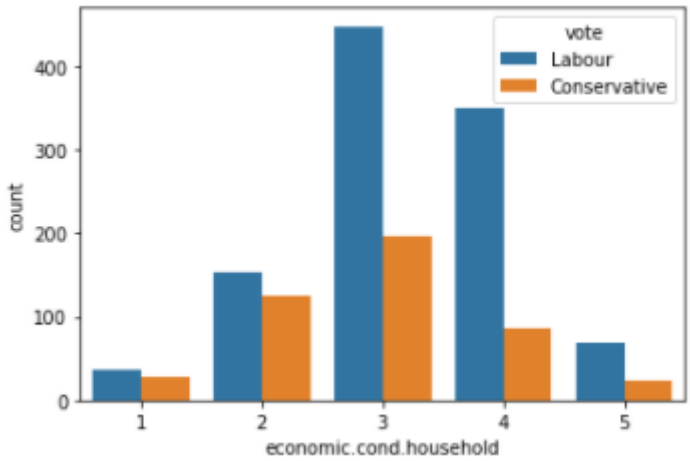


Fig.29

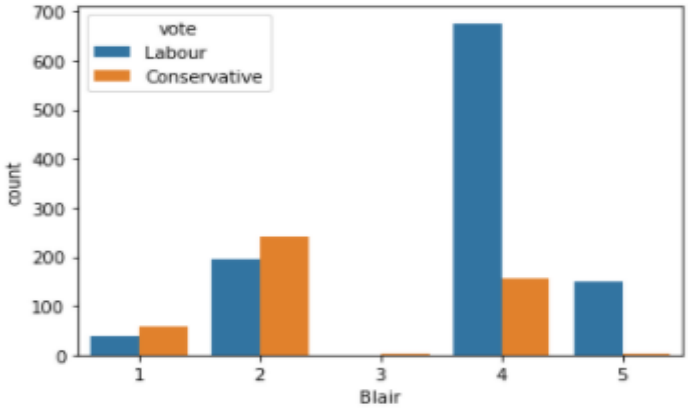


Fig.30

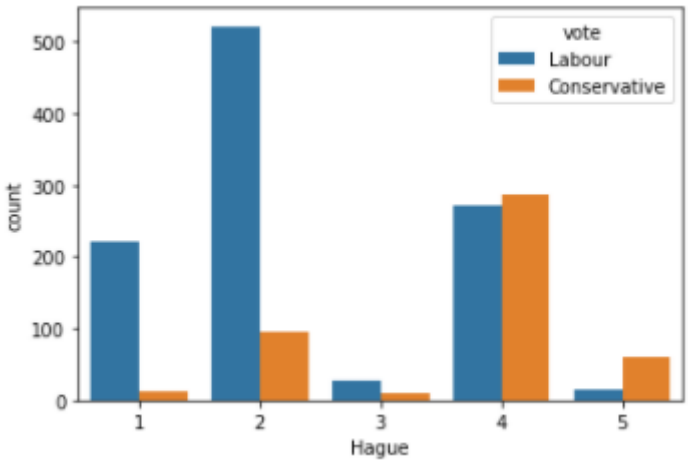


Fig.31

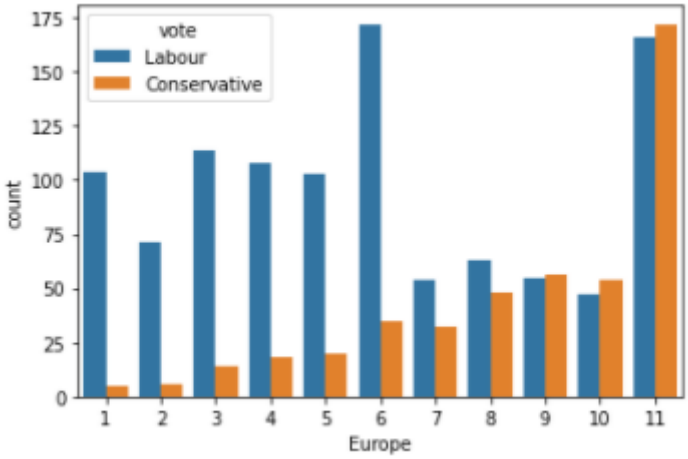


Fig.32

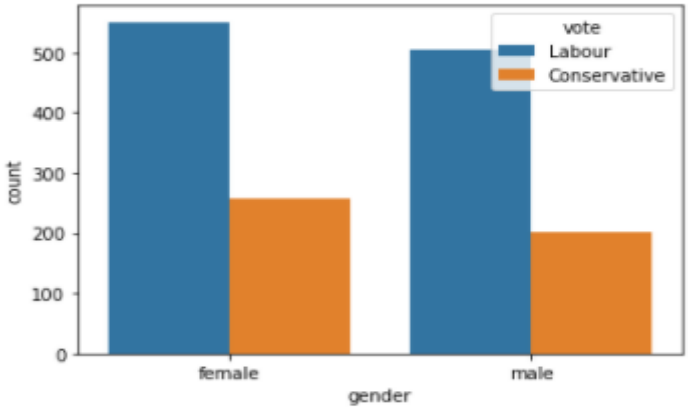


Fig.33

**Inferences:**

- Using the. type's function, we can check the type of data in each variable.
- Object means it is string or a combination of string and number or special characters. Integers take numerical value – 1,2, etc.
- In the table below, as we see that Vote and Gender both are of Object types.
- Vote has two classes – Labour and Conservative. Gender has classes – Male and Female.
- We need to convert the object type variable to numerical types in order to apply various models. We will be using categorical coding method to encode the data into numerical format.

Inference on Outliers:

- We notice that variables 'Economic Condition Household' and 'Economic condition National' have outliers present in the boxplot graph above. However, upon further investigating the data, we notice that all the values in these two columns are ordinal in nature.
- They follow a certain order, 1 being the least and 5 being the maximum.
- In general, ordinal data All other variables have no outliers present.

Conclusion:

- Values in both the columns – 'Economic Condition Household' and 'Economic condition National' are of ordinal type, i.e. they follow a certain order or degree of magnitude.
- The general assessments of current Economic Condition for Household and National have been rated '3' and '4'
- depicting a better overall economic condition. We can thus check the policies, laws and regulations of the party which will accentuate the economic growth basis certain economic factors.
- More number of voters have assessed Blair higher i.e., given him a '4' rating whereas a higher number of voters have assessed Hague on the lower side as that of '2'.
- One plausible explanation as depicted from the graph above is that the those who have voted for Labour party have assessed Blair, leader of the Labour party, a higher rating and the same voters would have assessed Hague,
- the leader of Conservative party a lower rating – this can be seen from the blue bar with highest count of over 500 who have assessed Hague on a scale '2' out of 5.
- People who have higher Eurosceptic sentiments tend to prefer Conservative Party whereas those with lower Eurosceptic sentiments are the ones who prefer Labour Party.
- More number of females have voted for Labour Party than Male members. Also, overall female voters are higher in number than male voters.

Observing the histograms of the variables, we can conclude:

- Distribution of "age" resembles normal distribution and is slightly right skewed.
- Distribution of "economic.cond.national" is not normal and it is slightly left skewed.
- Distribution of "economic.cond.household" is not normal and it is slightly left skewed.
- Distribution of "Blair" is not normal and it is slightly left skewed.

- Distribution of “Hague” is not normal and it is slightly right skewed.
- Distribution of “Europe” is somewhat normal and it is left skewed.
- Distribution of “political.knowledge” is not normal and it is left skewed.
- votes are large in number for ‘Labour’.
- ‘female’ voters large in number than ‘male’.

Age:

- Mean of the age variable is greater than the median
- It is noticed that majority of participants are within age group of 30 to 80 years
- Age group 45 to 50 are highest number of participants i.e. around 220 people.
- The survey has covered participants wide spread from age group 25 to 95 years.

economic.cond.national:

- Out of the 1525 participants around 600 participants rated the national economic condition as more than average (i.e scale of 3-3.3).

economic.cond.household:

- Out of the 1525 participants around 650 participants rated the household economic
- condition as more than average (i.e. scale of 3-3.4).

Blair:

- Blair is the labour leader.
- Approximately 850 number of participants has given above average Assessment of 3.7 to 4.3

Hague:

- Hague belongs to conservative party.
- Approximately 625 number of participants has given above average Assessment of 1.7 to 2.3

Political Knowledge:

- Both the party’s has good knowledge on European political exposure.

Europe:

- A Eurosceptic is a person, especially a politician, who is opposed to closer links between Britain and the European Union.
- Here, the majority of the respondents fall under the highest range, where they are Eurosceptic.

- There is not much outliers for the continuous variables.

- Only economic_cond_household and economic_cond_national columns have one outlier.
- No. of males who voted are 713 as against female of 812.
- Labour gets the highest voting from both female and male voters.
- Almost in all the categories Labour is getting the maximum votes.
- Conservative gets a little bit high votes from Europe 11.
- People who voted conservative are older and Europe 1 are older people.

**Multivariate Analysis:**

Correlation:


Fig.34

Correlation Data frame:

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

Fig.35

**Pair plot:**

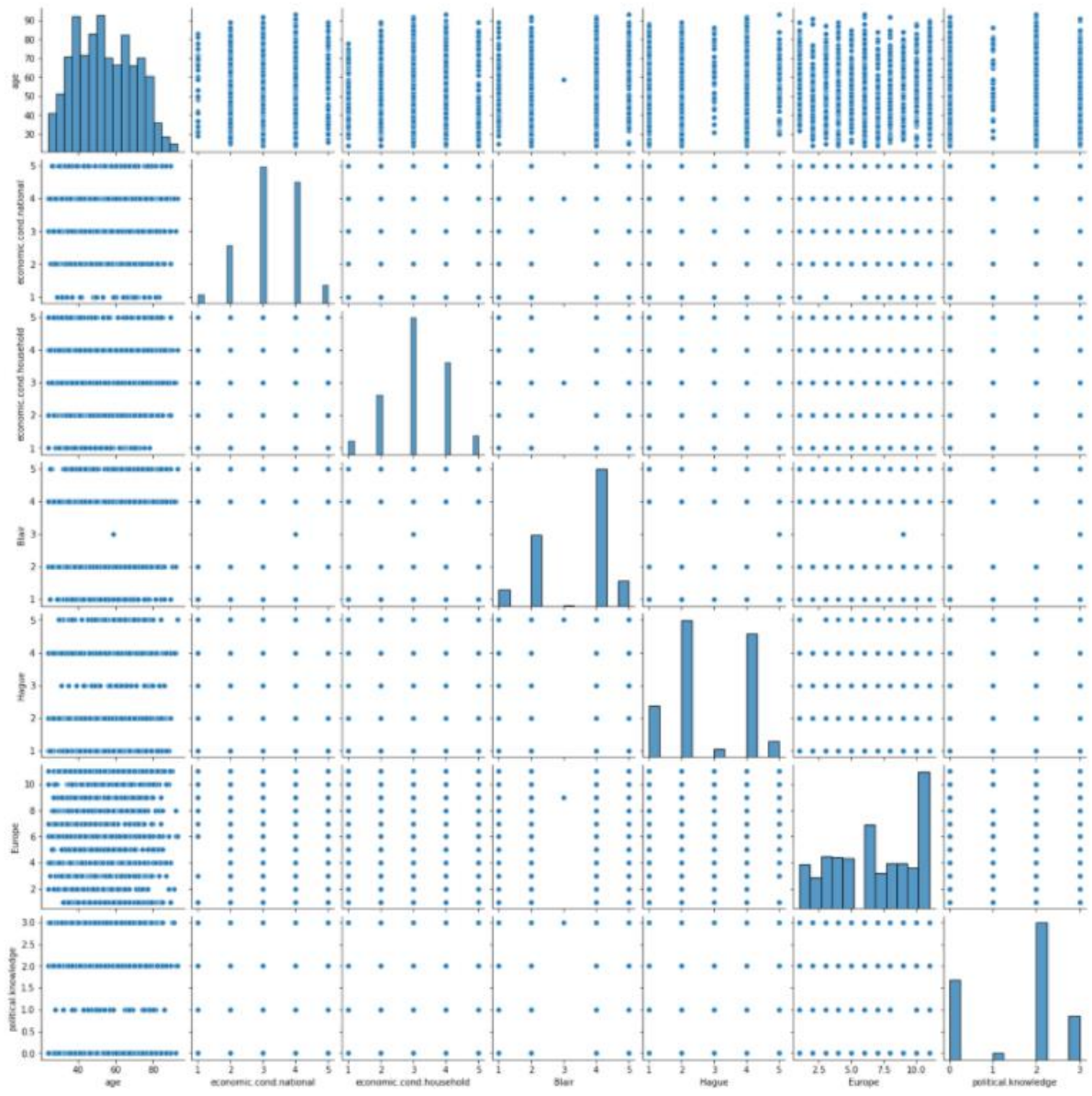


Fig.36

Inferences:

We can draw below inferences from multivariate analysis:

- There is hardly any collinear relationship among the variables.
- National and household economic condition have a weak positive correlation.
- Voters who rate national economic condition as high has a weak tendency to favour Labour party.
- There is negative correlation between age and & political knowledge
- High correlation between economic.cond.household and economic.cond.national(0.35)
- Followed by economic.cond.national and Blair(0.33)
- Proceeded by Europe and Hague (0.29)
- Voters who are Eurosceptic weakly tend to favour Conservative party.
- Voters who are not Eurosceptic weakly tend to favour Labour party.

**1.3 Encode the data (having string values) for Modelling. Is Scaling necessary here or not? Data Split: Split the data into train and test (70:30).**

- The variables 'vote' and 'gender' have string values. Converting them into numeric values for modelling
- We have encoded categorical variables vote and age by giving 1,0 to Labour and conservative respectively
- 0,1 to female and male respectively.

| | age | economic.cond.national | economic.cond.household | Blair | Hague | Europe | political.knowledge | vote_Labour | gender_male |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 43 | 3 | 3 | 4 | 1 | 2 | 2 | 1 | 0 |
| 1 | 36 | 4 | 4 | 4 | 4 | 5 | 2 | 1 | 1 |
| 2 | 35 | 4 | 4 | 5 | 2 | 3 | 2 | 1 | 1 |
| 3 | 24 | 4 | 2 | 2 | 1 | 4 | 0 | 1 | 0 |
| 4 | 41 | 2 | 2 | 1 | 1 | 6 | 2 | 1 | 1 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1520 | 67 | 5 | 3 | 2 | 4 | 11 | 3 | 0 | 1 |
| 1521 | 73 | 2 | 2 | 4 | 4 | 8 | 2 | 0 | 1 |
| 1522 | 37 | 3 | 3 | 5 | 4 | 2 | 2 | 1 | 1 |
| 1523 | 61 | 3 | 3 | 1 | 4 | 11 | 2 | 0 | 1 |
| 1524 | 74 | 2 | 3 | 2 | 4 | 11 | 0 | 0 | 0 |

Fig.37

- We are not going to scale the data for Logistic regression, LDA and Naive Bayes' models as it is not necessary.
- But in case of KNN it is necessary to scale the data, as it a distance-based algorithm (typically based on Euclidean distance).
- Gender variable is encoded using one hot encoding.
- The target variable, vote, is label encoded.

- Labour: 1
- Conservative: 0
- Here, even though the variables are not all at scale, we need not perform scaling, as the variables are ordinal in nature. Ordinal natured variables need not be scaled for modelling.
- We split the data into train and test set in a 70:30 ratio to perform further analysis and building our machine learning models.
- Since Vote and Age are categorical data, we encode the data using categorical coding.
- The following table shows data which take values 0 and 1 for column 'Vote' and 'Gender'
- We can see that the count of class 0 and class 1 in vote is 465 and 1061 respectively, where as in Gender, females' voters (808) are more in number than males' voters (709).
- Train Test Split:
- We split the data into train and test set in the ratio 70:30 where 70% of our data (i.e. 1061 observations) will be used for training purposes and 30% (i.e. 456 observations) will be used for testing purposes.
- X = all independent variables ['age', 'economic.cond.national', 'economic.cond.household', 'Blair', 'Hague', 'Europe', 'political.knowledge', 'gender']
- Y = dependent variable ['vote']
- We see in the table below that variable 'vote' has been dropped from X and it only takes independent variables.

```
Number of rows and columns of the training set for the independent variables: (1061, 8)
Number of rows and columns of the training set for the dependent variables: (1061,)
Number of rows and columns of the testing set for the independent variables: (456, 8)
Number of rows and columns of the testing set for the dependent variables: (456,)
```

Fig.38

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

**Logistic Regression:**

Applying Logistic Regression and fitting the training data:

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

Fig.39

Predicting train and test:

| | 0 | 1 |
|---|---|---|
| 0 | 0.931825 | 0.068175 |
| 1 | 0.096984 | 0.903016 |
| 2 | 0.298416 | 0.701584 |
| 3 | 0.110210 | 0.889790 |
| 4 | 0.017223 | 0.982777 |

Fig.40

| | 0 | 1 |
|---|---|---|
| 0 | 0.424284 | 0.575716 |
| 1 | 0.148426 | 0.851574 |
| 2 | 0.007187 | 0.992813 |
| 3 | 0.836350 | 0.163650 |
| 4 | 0.068407 | 0.931593 |

Fig.41

Train and Test accuracy:

0.8312912346842601 (train accuracy)
0.8355263157894737 (test accuracy)

Confusion matrix and Classification report for Train:

```
[[196 111]
 [ 68 686]]
              precision    recall  f1-score   support

           0       0.74      0.64      0.69       307
           1       0.86      0.91      0.88       754

    accuracy                           0.83      1061
   macro avg       0.80      0.77      0.79      1061
weighted avg       0.83      0.83      0.83      1061
```

Fig.42

Confusion matrix and Classification report for Test:

```
[[113  40]
 [ 35 268]]
              precision    recall  f1-score   support

           0       0.76      0.74      0.75       153
           1       0.87      0.88      0.88       303

    accuracy                           0.84       456
   macro avg       0.82      0.81      0.81       456
weighted avg       0.83      0.84      0.83       456
```

Fig.43

AUC ROC curve for Logistic Regression Train:

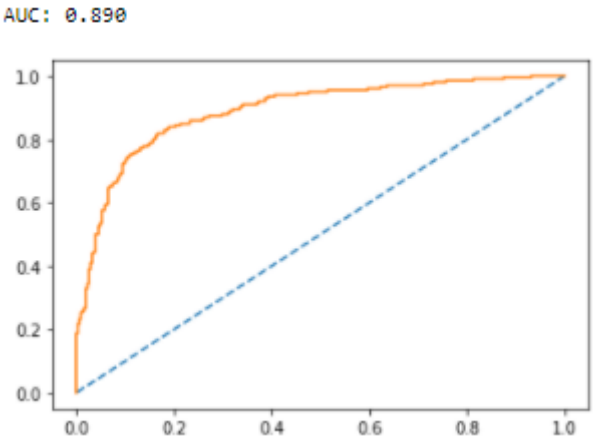


Fig.44

AUC ROC curve for Logistic Regression Test:

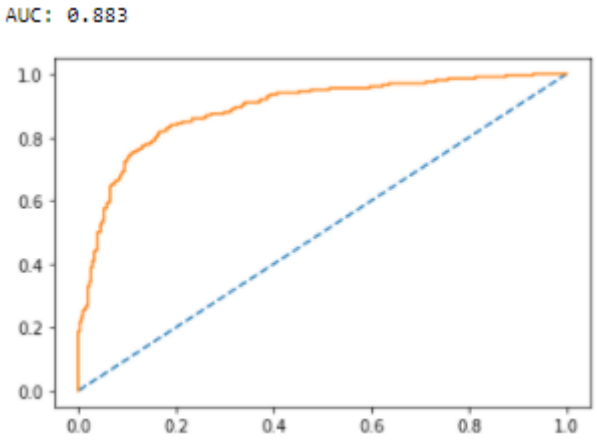


Fig.45

- The model is not overfitting or underfitting. Training and Testing results show that the model is excellent with good precision and recall values.

**LDA (linear discriminant analysis):**

Train and Test accuracy:

0.8341187558906692 (train accuracy)
0.8333333333333334 (test accuracy)

Confusion matrix and Classification report for Train:

```
[[200 107]
 [ 69 685]]
              precision    recall  f1-score   support

           0       0.74      0.65      0.69       307
           1       0.86      0.91      0.89       754

    accuracy                           0.83      1061
   macro avg       0.80      0.78      0.79      1061
weighted avg       0.83      0.83      0.83      1061
```

Fig.46

Confusion matrix and Classification report for Test:

```
[[111  42]
 [ 34 269]]
              precision    recall  f1-score   support

           0       0.77      0.73      0.74       153
           1       0.86      0.89      0.88       303

    accuracy                           0.83       456
   macro avg       0.82      0.81      0.81       456
weighted avg       0.83      0.83      0.83       456
```

Fig.47

AUC ROC curve for LDA on Train:

```
the auc 0.889
```

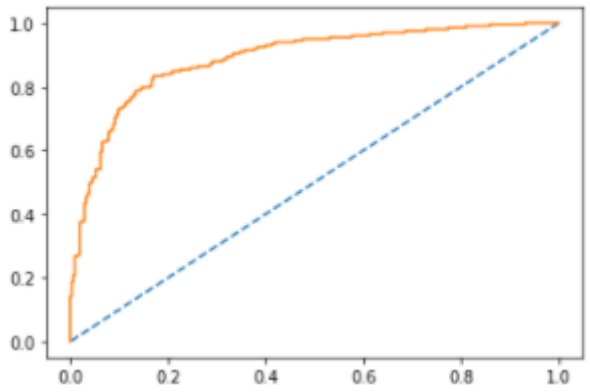


Fig.48

AUC ROC curve for LDA Test:

```
the auc curve 0.888
[<matplotlib.lines.Line2D at 0x19f2967a3a0>]
```

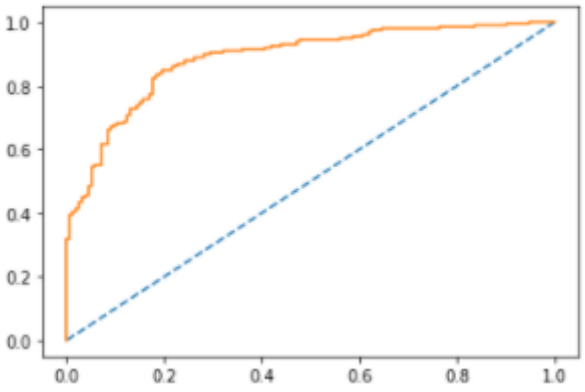


Fig.49

- Training and Testing results show that the model is excellent with good precision and recall values.
- Logistic Regression and LDA models have performed well in both the Training and Test data.
- While the model results between training and test sets are similar, indicating no under or overfitting issues.

Labour:

- AUC is 88% for both Training and Test data for logistic regression model
- LDA AUC for training is 88% while for Test is 88%

- Recall and Precision is high for both the data

Conservative:

- Recall, F1 score and Precision is less when compared to Labour party

**Inferences:**

Logistic Regression Model is a supervised learning classification algorithm used to predict the probability of a tar

get variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

In the given dataset, the target variable Vote has two classes – Labour (1) and Conservative (0).

Linear Discriminant Analysis is a dimensionality reduction technique which is commonly used for the supervised classification problems.

It is used for modelling differences in groups i.e., separating two or more classes.

It is most often used when the criterion or the dependent variable is categorical and the predictor or the

independent variable is interval in nature. In the given dataset, the dependent variable is categorical whereas independent variables take values.

LDA = LinearDiscriminantAnalysis(solver='lsqr', shrinkage=None, priors=None, n_components=-1,

store_covariance=False, tol=0.0001, covariance_estimator=None)

LDA_model = LDA.fit(X_train,y_train)

Interpreting the Inferences of both the Models – Logistic Regression & LDA

1. Logistic Regression is traditionally used for two-class and binary classification problems. Though it can be extra plated and used in multi-class classification, this is rarely performed.

On the other hand, Linear Discriminant A analysis is considered a better choice whenever multi-class classification is required and in the case of binary classifications , both logistic regression and LDA are applied.

In the election dataset, since it is a binary classification problem, both Logistic Regression and LDA have performed almost equally well.

2. Both models – Logistic Regression and LDA have performed better for class 1 than for class 0. To put it in other words – both the models are better at predicting voters who voted for Labour party than for Conservative party.

A plausible explanation for this could be that Class 1 is better represented with 1061 number of observations as against class 0 with 465 observations.

3. In the given dataset, there are fewer observations and variables from which the parameters are to be estimated. In such a case, logistic regression becomes unstable.

However, Linear Discriminant Analysis is a better option because it tends to be stable even in such cases.

4. Logistic Regression can lack stability when the classes are well-separated. This is where LDA comes in.

5. Accuracy of the model is of greater importance in the given dataset, the reason being that one needs to predict both the classes 0 & 1 correctly. We will check for model that has higher number of correct predictions

- Logistic Regression Model Score on Train Data: 0.8312912346842601

- Logistic Regression Model Score on Test Data: 0.8355263157894737

- LDA model score for train data is: 0.8341187558906692 - LDA model score for test data is: 0.8333333333333334

6. The number of correct predictions for Logistic Regression model for Train and Test data is 885 & 383 respectively whereas for LDA model is 882 and 382.

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

**KNN Model:**

Applying KNN model and fitting the training data:

```
KNeighborsClassifier()
```

Fig.50

Train and Test accuracy:

0.8566402814423922 (train accuracy)
0.8263157894736842 (test accuracy)

Confusion matrix and Classification report for Train:

```
[[232  95]
 [ 68 742]]
              precision    recall  f1-score   support

           0       0.77      0.71      0.74       327
           1       0.89      0.92      0.90       810

    accuracy                           0.86      1137
   macro avg       0.83      0.81      0.82      1137
weighted avg       0.85      0.86      0.85      1137
```

Fig.51

Confusion matrix and Classification report for Test:

```
[[ 91  42]
 [ 24 223]]
              precision    recall  f1-score   support

           0       0.79      0.68      0.73       133
           1       0.84      0.90      0.87       247

    accuracy                           0.83       380
   macro avg       0.82      0.79      0.80       380
weighted avg       0.82      0.83      0.82       380
```

Fig.52

AUC ROC curve for KNN Train:

```
the auc 0.929
```

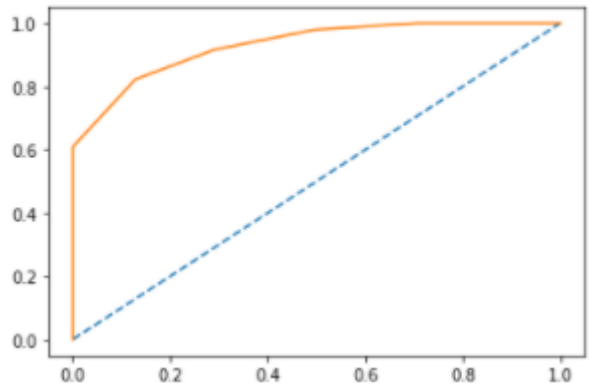


Fig.53

AUC ROC curve for KNN Test:

```
the auc curve 0.868

[<matplotlib.lines.Line2D at 0x19f282dda30>]
```

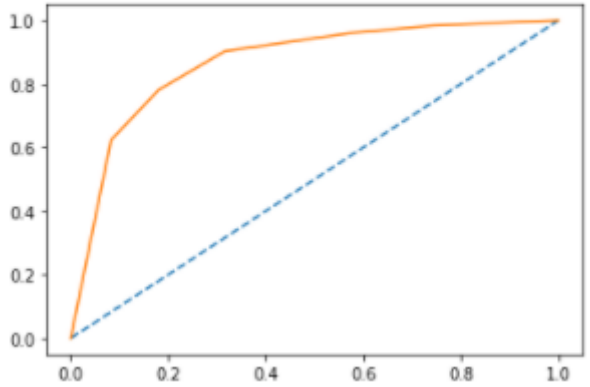


Fig.54

- The model is not overfitting or underfitting. Training and Testing results show that the model is excellent with good precision and recall values.

**Naïve Bayes:**

Applying Naïve Bayes model and fitting the training data:

```
GaussianNB()
```

Train and Test accuracy:

0.8350612629594723 (train accuracy)
0.8223684210526315 (test accuracy)

Confusion matrix and Classification report for Train:

```
[[211  96]
 [ 79 675]]
              precision    recall  f1-score   support

           0       0.73      0.69      0.71       307
           1       0.88      0.90      0.89       754

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

Fig.55

Confusion matrix and Classification report for Test:

```
[[112  41]
 [ 40 263]]
              precision    recall  f1-score   support

           0       0.74      0.73      0.73       153
           1       0.87      0.87      0.87       303

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

Fig.56

AUC ROC curve for Naïve Bayes on Train:

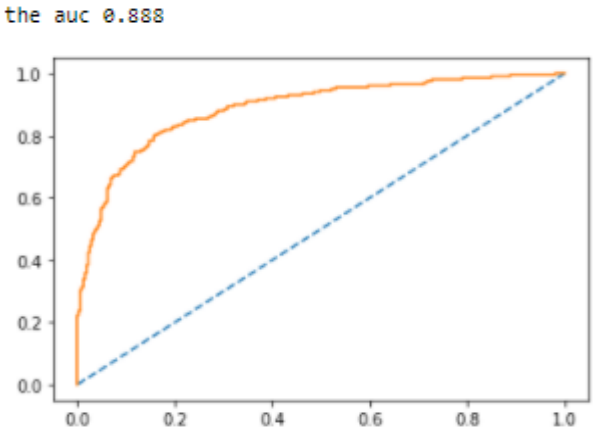


Fig.57

AUC ROC curve for Naïve Bayes Test:

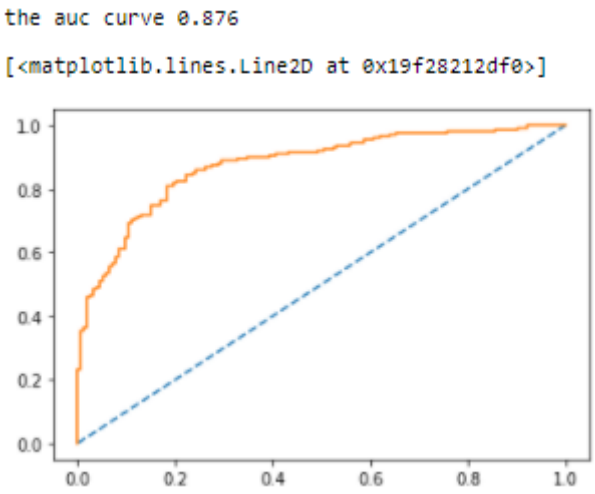


Fig.58

- Naïve bayes and KNN models have performed well in both the Training and Test data.
- While the model results between training and test sets are similar, indicating no under or overfitting issues.

Labour:

- AUC is 92% for both Training and Test data for KNN
- LDA AUC for training is 88% while for Test is 88%
- Precision is high for KNN Conservative-
- Recall, F1 score and Precision is less when compared to Labour party

Conservative:

- Recall, F1 score and Precision is less when compared to Labour party

**Inferences:**

One of the simplest Machine Learning algorithms based on supervised learning technique, K-Nearest Neighbours
(KNN) is used in for regression and classification problem. KNN algorithms use data and classify new data points based on similarity measures (e.g. distance function). Classification is done by a majority vote to its neighbours.
The data is assigned to the class which has the nearest neighbours. As you increase the number of nearest neighbours, the value of k, accuracy might increase.

Defined KNN model & Fit the data
KNN_model=KNeighborsClassifier (n_neighbors=7, weights='uniform', algorithm='auto',leaf_size=40, p=2, metric='minkowski', metric_params=None, n_jobs=None) KNN_model =KNN_model.fit(X_train,y_train)

Naive Bayes Model: Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a

common principle, i.e. every pair of features being classified is independent of each other. Define Naive Bayes model & Fit the data

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

Interpreting the Inferences of both the Models – KNN and Naïve Bayes
1. Accuracy: Naïve Bayes model has better accuracy on Test data than KNN for both Class 0 and Class 1 with an accuracy score of 0.84 as against 0.82 for Train data. We can thus state that Naïve Bayes is a better overall model between the two for predicting correct outcomes.
2. Naive Bayes is much faster than KNN due to KNN's real-time execution. Naive Bayes is parametric whereas KNN is non-parametric.
3. Precision, Recall and F1 Score is better for Test data set for Naïve Bayes Model than for KNN Model for both the classes. F1 score is a better metric since we are looking for correct predictions for both the class 0 and class 1. F1 score for Test data of NB model for class 0 is 0.73 whereas for class 1 is 0.89 which is higher than KNN model F1 scores for Test data for class 0 & 1.
4. We also notice that there is a class imbalance where class 1 (1061) has more representation than class 0 (465). In most real-life classification problems, imbalanced class distribution exists and thus F1-score is a better metric to evaluate our model on.

**1.6 Model Tuning (4 pts), Bagging (1.5 pts) and Boosting (1.5 pts). Apply grid search on each model (include all models) and make models on best_params. Define a logic behind choosing particular values for different hyper-parameters for grid search. Compare and comment on performances of all. Comment on feature importance if applicable. Successful implementation of both algorithms along with inferences and comments on the model performances.**

Model Tuning:

Tuning is the process of maximizing a model's performance without overfitting or creating too high of a variance. In machine learning, this is accomplished by selecting appropriate "hyperparameters"

Every model comes with a few hyperparameters that one can apply to optimise the model performance. The most widely used method for model tuning is GridSearchCV from sklearn model selection library.

For Logistic Regression Model, some of the hyper parameters used are as follows:

```
'penalty': ['l1','none'],
'solver': ['lbfgs','liblinear'],
'max_iter': [150, 200],
'multi_class':['ovr', 'auto'],
'tol':[0.0001,0.00001]}
```

For LDA model, hyperparameters used are:

GridSearchCV(cv=3, estimator=LinearDiscriminantAnalysis(), param_grid={'n_components': ['None', '-1'], 'shri nkage': ['None', 'auto'], 'solver': ['lsqr', 'svd']})

Similarly, we can change the hyperparameters to test which combination gives higher accuracy or model score. The model that gives the highest scores can be used as the best model. We have applied gridsearchcv on our models and changed the parameters to give us the best results.

Before we make the Bagging Model, Lets see how Random Forest Model Performs without Bagging and then we will compare both the Models

Random Forest Model:

Random forest is a supervised learning algorithm that can be used for classification and regression problems. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model.

RF_model=RandomForestClassifier(n_estimators=100, random_state=0) RF_model.fit(X_train, y_train)

Bagging Classifier:

A Bagging classifier is an ensemble meta-estimator that fits base classifiers such as decision trees or Random Forest each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. It is a way used to reduce the variance by introducing randomization into its construction procedure and then making an ensemble out of it. In this particular problem, we will use the base estimator as Random Forest and compare the evaluation metrics for the two models.

Defined Bagging model & Fit the data cart RandomForestClassifier(n_estimators=100,random_state=1)

Bagging_model=BaggingClassifier(base_estimator=cart,n_estimators=100,random_state=0) #we have been specifically asked to use Random Forest model Bagging_model.fit(X_train, y_train)

Bagging:

```
BaggingClassifier(base_estimator=RandomForestClassifier(), n_estimators=10
0,
                  random_state=1)
```

Fig.59

Performance Matrix on train data set:

```
0.9679547596606974
[[277  30]
 [  4 750]]
              precision    recall  f1-score   support

           0       0.99      0.90      0.94       307
           1       0.96      0.99      0.98       754

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

Fig.60

Performance Matrix on test data set:

```
0.8289473684210527
[[104  49]
 [ 29 274]]
              precision    recall  f1-score   support

           0       0.78      0.68      0.73       153
           1       0.85      0.90      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
```

Fig.61

Comparison of AUC and ROC for training and test data:

```
AUC for training data: 0.997
AUC for test data: 0.997
```

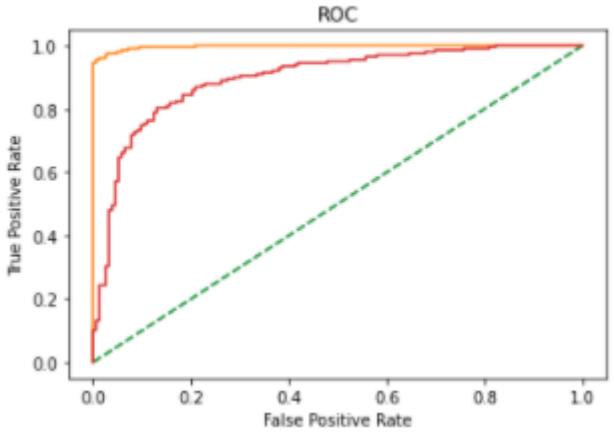


Fig.62

Boosting Model has two types – ADA Boost and Gradient Boost, we will check the model performance of both the types of Boosting models

The term 'Boosting' refers to a family of algorithms which converts weak learner to strong learners Boosting is used to create a collection of predictors. In this technique, learners are learned sequentially with early learners fitting simple models to the data and then analysing data for errors. Consecutive trees (random sample) are fit and at every step, the goal is to improve the accuracy from the prior tree.

Defined ADA Boost model & Fit the data

A type of ensemble technique where a number of weak learners are combined together to form a strong learner. Here, usually, each weak learner is developed as decision stumps (A stump is a tree with just a single split and two terminal nodes) that are used to classify the observations.

Adaboost increases the predictive accuracy by assigning weights to both observations at end of every tree and weights(scores) to every classifier. Hence, in Adaboost, every classifier has a different weightage on final prediction contrary to the random forest where all trees are assigned equal weights.

```
ADB_model = AdaBoostClassifier(n_estimators=100, random_state=0)
ADB_model.fit(X_train,y_train)
```

Ada Boost Classifier:

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

Performance Matrix on train data set:

```
0.8501413760603205
[[214  93]
 [ 66 688]]
              precision    recall  f1-score   support

           0       0.76      0.70      0.73       307
           1       0.88      0.91      0.90       754

    accuracy                           0.85      1061
   macro avg       0.82      0.80      0.81      1061
weighted avg       0.85      0.85      0.85      1061
```

Fig.63

Performance Matrix on test data set:

```
0.8135964912280702
[[103  50]
 [ 35 268]]
              precision    recall  f1-score   support

           0       0.75      0.67      0.71       153
           1       0.84      0.88      0.86       303

    accuracy                           0.81       456
   macro avg       0.79      0.78      0.79       456
weighted avg       0.81      0.81      0.81       456
```

Fig.64

Comparison of AUC and ROC for training and test data:

```
AUC for training data: 0.915
AUC for test data: 0.915
```

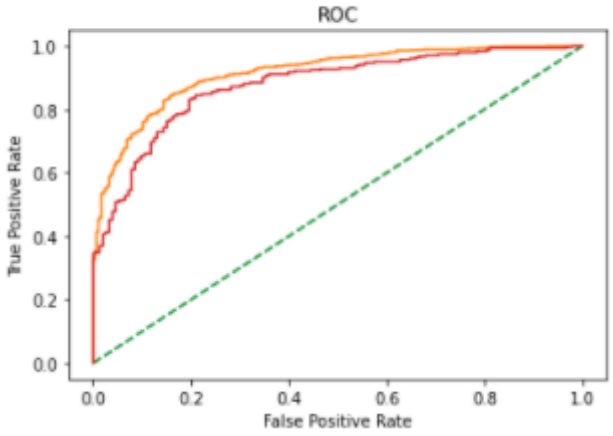


Fig.65

ADB model score:

```
Accuracy on training set :  0.8501413760603205
Accuracy on test set :  0.8135964912280702
Recall on training set :  0.9124668435013262
Recall on test set :  0.8844884488448845
Precision on training set :  0.8809218950064021
Precision on test set :  0.8427672955974843
```

Fig.66

Gradient Boosting Model

Defined Gradient Boosting model & fit the data

Gradient Boost Just like AdaBoost, Gradient Boost also combines a no. of weak learners to form a strong learner. Here, the residual of the current classifier becomes the input for the next consecutive classifier on which the trees are built, and hence it is an additive model. The residuals are captured in a step-by-step manner by the classifiers, in order to capture the maximum variance within the data, this is done by introducing the learning rate to the classifiers.

By this method, we are slowly inching in the right direction towards better prediction (This is done by identifying negative gradient and moving in the opposite direction to reduce the loss, hence it is called Gradient Boosting in line with Gradient Descent where similar logic is employed). Thus, by no. of classifiers, we arrive at a predictive value very close to the observed value.

Fitting the Model

GB_model= GradientBoostingClassifier(random_state=0) GB_model= GB_model.fit(X_train, y_train)

Gradient Boost Classifier:

```
GradientBoostingClassifier(random_state=1)
```

Performance Matrix on train data set:

```
0.8925541941564562
[[239  68]
 [ 46 708]]
              precision    recall  f1-score   support

           0       0.84      0.78      0.81       307
           1       0.91      0.94      0.93       754

    accuracy                           0.89      1061
   macro avg       0.88      0.86      0.87      1061
weighted avg       0.89      0.89      0.89      1061
```

Fig.67

Performance Matrix on test data set:

```
0.8355263157894737
[[105  48]
 [ 27 276]]
              precision    recall  f1-score   support

           0       0.80      0.69      0.74       153
           1       0.85      0.91      0.88       303

    accuracy                           0.84       456
   macro avg       0.82      0.80      0.81       456
weighted avg       0.83      0.84      0.83       456
```

Fig.68

Comparison of AUC and ROC for training and test data:

```
AUC for training data: 0.951
AUC for test data: 0.951
```

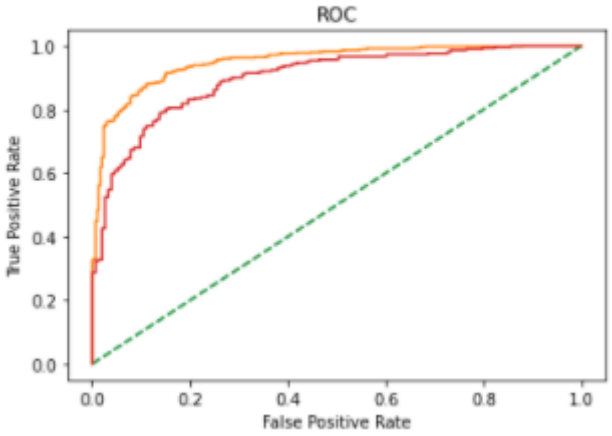


Fig.69

ADB model score:

```
Accuracy on training set :  0.8925541941564562
Accuracy on test set :  0.8355263157894737
Recall on training set :  0.9389920424403183
Recall on test set :  0.9108910891089109
Precision on training set :  0.9123711340206185
Precision on test set :  0.8518518518518519
```

Fig.70

**Hyperparameter Tuning**

Applying GridSearchCV for Logistic Regression :

Logistic Regression tuned:

```
LogisticRegression(max_iter=10000, n_jobs=2)
```

Grid search:

```
GridSearchCV(cv=3, estimator=LogisticRegression(max_iter=10000, n_jobs=2),
             n_jobs=-1,
             param_grid={'penalty': ['l2', 'none'], 'solver': ['sag', 'lbfg
s'],
                         'tol': [0.0001, 1e-05]},
             scoring='f1')
```

Fig.71

```
{'penalty': 'l2', 'solver': 'sag', 'tol': 1e-05}
```

```
LogisticRegression(max_iter=10000, n_jobs=2, solver='sag', tol=1e-05)
```

Fig.72

Performance Matrix on train data set:

```
0.8322337417530632
[[196 111]
 [ 67 687]]
              precision    recall  f1-score   support

           0       0.75      0.64      0.69       307
           1       0.86      0.91      0.89       754

    accuracy                           0.83      1061
   macro avg       0.80      0.77      0.79      1061
weighted avg       0.83      0.83      0.83      1061
```

Fig.73

Performance Matrix on test data set:

```
0.8267543859649122
[[110  43]
 [ 36 267]]
              precision    recall  f1-score   support

           0       0.75      0.72      0.74       153
           1       0.86      0.88      0.87       303

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

Fig.74

LR model score:

```
Accuracy on training set :  0.8312912346842601
Accuracy on test set :  0.8355263157894737
Recall on training set :  0.9098143236074271
Recall on test set :  0.8844884488448845
Precision on training set :  0.8607277289836889
Precision on test set :  0.8701298701298701
```

Fig.75

LR model tuned:

```
Accuracy on training set :  0.8312912346842601
Accuracy on test set :  0.8355263157894737
Recall on training set :  0.9098143236074271
Recall on test set :  0.8844884488448845
Precision on training set :  0.8607277289836889
Precision on test set :  0.8701298701298701
```

Fig.76

Model tuning on KNN:

```
Accuracy Score for K=3 is  0.7894736842105263
Accuracy Score for K=5 is  0.8157894736842105
Accuracy Score for K=9 is  0.8048245614035088
```

Fig.77

predicting the response:

```
Accuracy Score for K=3 is  0.8048245614035088
Accuracy Score for K=5 is  0.8157894736842105
Accuracy Score for K=9 is  0.8048245614035088
```

Fig.78

Empty list that will hold accuracy scores:

```
[0.2192982456140351,
 0.21052631578947367,
 0.1842105263157895,
 0.17324561403508776,
 0.19517543859649122,
 0.19956140350877194,
 0.19956140350877194,
 0.1907894736842105,
 0.20394736842105265,
 0.19736842105263153]
```

Fig.79

plotting misclassification error vs k:

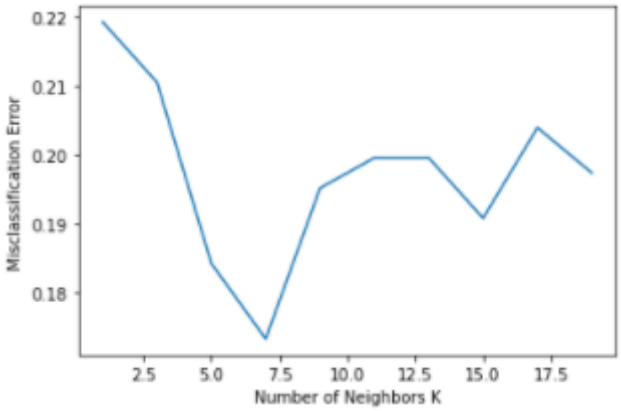


Fig.80

Model Tuning on Adaboost Classifier:

```
AdaBoostClassifier(random_state=1)
```

ADB model score:

```
Accuracy on training set :  0.8501413760603205
Accuracy on test set :  0.8135964912280702
Recall on training set :  0.9124668435013262
Recall on test set :  0.8844884488448845
Precision on training set :  0.8809218950064021
Precision on test set :  0.8427672955974843
```

Fig.81

ADB model tuned:

```
Accuracy on training set :  0.8463713477851084
Accuracy on test set :  0.8135964912280702
Recall on training set :  0.9124668435013262
Recall on test set :  0.8778877887788779
Precision on training set :  0.8764331210191083
Precision on test set :  0.8471337579617835
```

Fig.82

ADB model feature importance's:

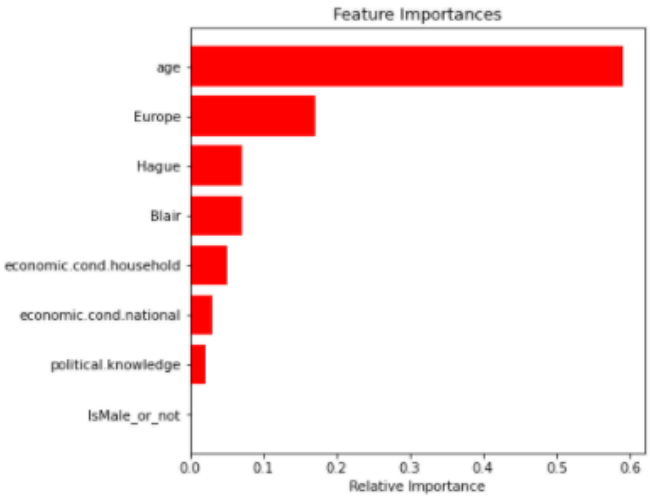


Fig.83

ADB tuned feature importance's:

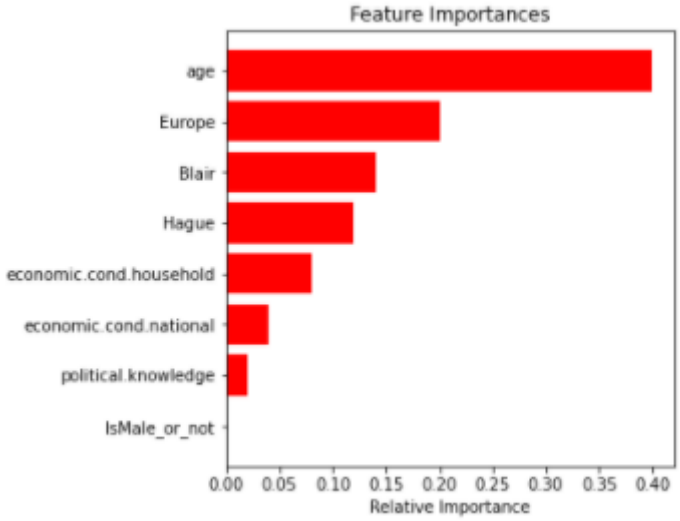


Fig.84

Model Tuning on Gradient Boost Classifier:

```
GradientBoostingClassifier(init=AdaBoostClassifier(random_state=1),
                           max_features=1, random_state=1, subsample=1)
```

Fig.85

Gradient Boost Classifier model tuned:

```
Accuracy on training set :  0.8737040527803959
Accuracy on test set :  0.8333333333333334
Recall on training set :  0.9350132625994695
Recall on test set :  0.9075907590759076
Precision on training set :  0.8924050632911392
Precision on test set :  0.8513931888544891
```

Fig.86

Gradient Boost Classifier tuned feature importance's:

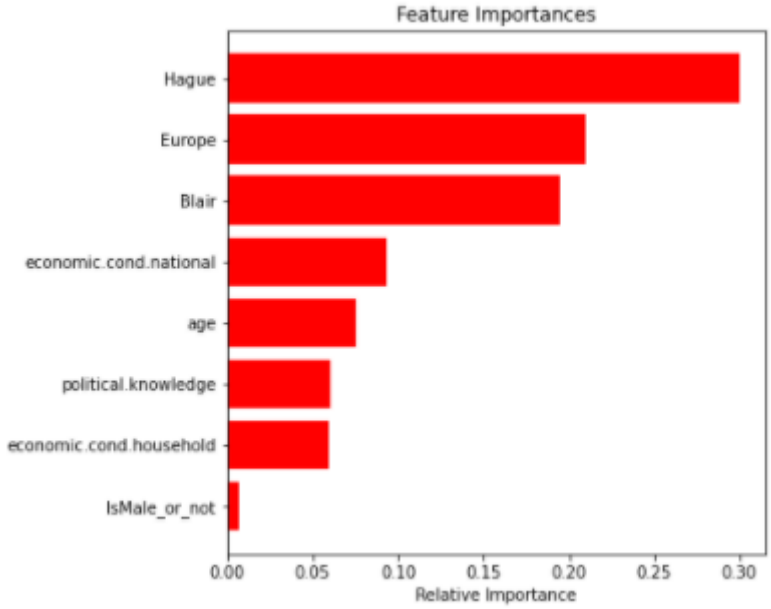


Fig.87

Inference:

1. Random Forest has performed exceptionally well on the train data for both the classes 0 and 1 with an accuracy score of 1. Recall, Precision, F1 score is also 1 on Train Data for both the classes.

However, a closer look at the metrics of the Testing Dataset shows that this model is highly overfitted.

2. On Testing Dataset for both the classes 0 and 1, the efficacy if the model reduces and with difference in scores for train and test data is higher.

RF Model predicts better for class 1 than for class 0 for Test data with an F1 Score of 0.88 for class 1 as against 0.71 for class 0 – clearly an indicator of class imbalance.

3. However, bagging classifier, on the other hand reduces the variance in scores for train data and Test data.

The F1 score is 0.98 for Train data as against 0.89 for Test Data for class 1 and for class0 – it is 0.95 and 0.70 for train and test data respectively.

4. For optimal Model performance we can apply hyperparameters to remove class imbalance and check if the performance of the model improves using bagging.

5. Changed some of the hyperparameters to finetune our model for greater accuracy.

Gradient Boost model has clearly performed better than ADA Boost Model on all the metrics:

1. Accuracy: For GB Model Train and Test data is 0.89 and 0.85 respectively as against 0.85 and 0.83 for ADA Boost Model for both Class 0 and Class 1 suggesting a better model performance.

2. AUC Score: For GB model is 0.95 and 0.90 for train and test data as against 0.91 and 0.88 for ADA boost train and test data.

3. F1 Score: is higher for GB Model than for ADA Boost Model

**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, classification report (4 pts) Final Model - Compare and comment on all models on the basis of the performance metrics in a structured tabular manner. Describe on which model is best/optimized, after comparison which model suits the best for the problem in hand on the basis of different measures. Comment on the final model.**

# Logistic Regression

Mean scores of LR model CV:

```
0.8312995944277907
```

Train scores of LR model CV:

```
array([0.82242991, 0.77358491, 0.83962264, 0.86792453, 0.85849057,
       0.8490566 , 0.81132075, 0.8490566 , 0.81132075, 0.83018868])
```

Test scores of LR model CV:

```
array([0.80434783, 0.7826087 , 0.84782609, 0.82608696, 0.89130435,
       0.86956522, 0.93333333, 0.84444444, 0.73333333, 0.82222222])
```

# Linear Discriminant Analysis:

Mean scores of LDR model CV and array:

```
0.8256656674307882
```

Train scores of LDR model CV

```
array([0.79439252, 0.77358491, 0.83962264, 0.85849057, 0.85849057,
       0.8490566 , 0.80188679, 0.8490566 , 0.81132075, 0.82075472])
```

Test scores of LDR model CV:

```
array([0.80434783, 0.7826087 , 0.84782609, 0.82608696, 0.89130435,
       0.86956522, 0.93333333, 0.84444444, 0.73333333, 0.82222222])
```

# KNN Model:

Mean scores of KNN model CV and array:

```
0.7766002468700406
```

```
0.7945247751719273
```

Train scores of KNN model CV

```
array([0.80373832, 0.78301887, 0.76415094, 0.80188679, 0.82075472,
       0.81132075, 0.79245283, 0.82075472, 0.77358491, 0.77358491])
```

Test scores of KNN model CV:

```
array([0.76086957, 0.82608696, 0.80434783, 0.7173913 , 0.82608696,
       0.82608696, 0.82222222, 0.8       , 0.75555556, 0.73333333])
```

# Naive Bayes Model:

Mean scores of NB model CV and array:

```
0.8284870393228706
```

Train scores of NB model CV

```
array([0.80373832, 0.78301887, 0.8490566 , 0.83962264, 0.90566038,
       0.8490566 , 0.78301887, 0.83962264, 0.81132075, 0.82075472])
```

Test scores of NB model CV:

```
array([0.7826087 , 0.80434783, 0.86956522, 0.80434783, 0.86956522,
       0.86956522, 0.88888889, 0.82222222, 0.75555556, 0.82222222])
```

# Bagging:

Mean scores of Bagging model CV and array:

```
0.8322253570798802
```

Train scores of Bagging model CV

```
array([0.8411215 , 0.81132075, 0.85849057, 0.8490566 , 0.90566038,
       0.79245283, 0.83018868, 0.81132075, 0.79245283, 0.83018868])
```

Test scores of Bagging model CV:

```
array([0.7826087 , 0.76086957, 0.84782609, 0.73913043, 0.86956522,
       0.84782609, 0.84444444, 0.8       , 0.73333333, 0.77777778])
```

# Ada Boost Classifier:

Mean scores of Ada Boost Classifier model CV and array:

```
0.8237700581907952
```

Train scores of Ada Boost Classifier model CV

```
array([0.80373832, 0.79245283, 0.83018868, 0.85849057, 0.89622642,
       0.81132075, 0.78301887, 0.83018868, 0.78301887, 0.8490566 ])
```

Test scores of Ada Boost Classifier model CV:

```
array([0.7173913 , 0.7173913 , 0.80434783, 0.69565217, 0.86956522,
       0.82608696, 0.91111111, 0.75555556, 0.68888889, 0.8       ])
```

# Gradient Boost Classifier:

Mean scores of Gradient Boost Classifier model CV and array:

```
0.8312731440663024
```

Train scores of Gradient Boost Classifier model CV

```
array([0.85046729, 0.80188679, 0.81132075, 0.8490566 , 0.88679245,
       0.83962264, 0.81132075, 0.83962264, 0.76415094, 0.85849057])
```

Test scores of Gradient Boost Classifier model CV:

```
array([0.73913043, 0.73913043, 0.86956522, 0.73913043, 0.82608696,
       0.84782609, 0.86666667, 0.82222222, 0.68888889, 0.75555556])
```

**# Comparing all Models:**

| | Model | Train_Accuracy | Test_Accuracy | Train_Recall | Test_Recall | Train_Precision | Test_Precision |
|---|---|---|---|---|---|---|---|
| 0 | Logistic Regression with default parameters | 0.83 | 0.84 | 0.91 | 0.88 | 0.86 | 0.87 |
| 1 | Logistic Regression Tuned | 0.83 | 0.84 | 0.91 | 0.88 | 0.86 | 0.87 |
| 2 | LDA Model | 0.83 | 0.83 | 0.91 | 0.89 | 0.86 | 0.86 |
| 3 | KNN Model | 0.85 | 0.80 | 0.93 | 0.89 | 0.87 | 0.83 |
| 4 | Naive Bayes Model | 0.84 | 0.82 | 0.90 | 0.87 | 0.88 | 0.87 |
| 5 | Bagging Model | 0.97 | 0.83 | 0.99 | 0.90 | 0.96 | 0.85 |
| 6 | Adaboost Model with default parameters | 0.85 | 0.81 | 0.91 | 0.88 | 0.88 | 0.84 |
| 7 | Adaboost Model Tuned | 0.85 | 0.81 | 0.91 | 0.88 | 0.88 | 0.85 |
| 8 | Gradient Boost Model with default parameters | 0.89 | 0.84 | 0.94 | 0.91 | 0.91 | 0.85 |
| 9 | Gradient Boost Model Tuned | 0.87 | 0.83 | 0.94 | 0.91 | 0.89 | 0.85 |

Fig.53

Inferences:

LDA: -

Labour:

- There is not much improvement in performance for the LDA model after hyper tuning.
- Hyper tuned model and Basic model has similar scores for Accuracy, Precision, Recall, F1score and AUC.
- After applying other hyperparameters, the test data has performed less when compared to Hyper tuned or Basic model
- SMOTE as a technique is applied only if minority class is between 1-2%. In our example minority class is 30%, hence for such data set, there is no need of oversampling.

Conservative:

- Hyper tuned and Basic model has similar score. But the values have increased after applying other hyperparameters. Even then all the parameter score is less than Labour Class

Logistic Regression Model: -

Labour:

- There is not much improvement in performance for the LR model after hyper tuning and other hyperparameters.
- Hyper tuned model and Basic model has similar scores for Accuracy, Precision and AUC.
- After applying other hyperparameters, the test data has performed less when compared to Hyper tuned or Basic model. Only Precision of Test data has comparatively increased.

Conservative:

- Hyper tuned and Basic model has similar score. But the values have increased after applying other hyperparameters. Even then the parameter score is less than Labour Class

KNN Model: -

Labour:

- In the Basic model, Accuracy and Precision, F1 Score is better than the Tuned model.
- Recall and AUC value for Test data is slightly higher in Tuned model.
- In applying other parameters, only Precision value has increased.
- The performance of test data does not improve much after hyper tuning.

Conservative:

- Basic model values are much better for Precision and F1 score.
- Tuned model, recall value is higher
- In other parameters, it is overfitted one as performed poorly on Test data

Naïve Bayes Model: –

labour:

- There is not much improvement in performance for the NB model after hyper tuning.
- Hyper tuned model and Basic model has similar scores for Accuracy, Precision, Recall, F1score and AUC.

Conservative:

- Hyper tuned model has not much increased the performance of Test data.

- Other models have increased the parameter values but they are still lesser than Labour Class scores

Bagging: -

labour:

- Though the Parameter score is good for Tuned model, it is overfitted one as it has not performed well in Test data.
- There is approx. 10% difference between Training and Test data and the overfitted issue is much better than Tuned model

Conservative:

- The Tuned model is also overfitted ones as it has performed poorly on Test data

ADAboosting and Gradient Boosting: –

Labour:

- Tuned model is better for ADA performance of Test data have increased - Accuracy, Precision, Recall, F1 score and AUC is higher
- For other models for ADA, performance has not much increased for Test data.
- For GB tuned model – It has performed approx. 100% in Training data but not well in the test data.
- Hence it is overfitted one. Basic model of GB is better.

Conservative:

- For ADA, performance has not much increased for Tuned model.
- Similarly, Basic model is better values for GB model where Tuned model is overfitted one and still values are lesser than Labour party class.

<u>Model evaluation metrics for class 1 – Labour:</u>

1. Variance in Scores:

Logistic Regression Model & LDA model have both performed uniformly on both Train and Test Data, with very little variance in the train and test data scores across various metrics. However, in terms of accuracy if falls short in front of other models such as Bagging and Gradient Boost which have higher accuracy

2. High Metrics:

Bagging Model has the highest scores across all metrics – F1, recall, precision and accuracy for training data. However, we cannot say the same for testing data – again showing high

variance in scores in train and test data. This also suggest that the model is overfitted. Once can further finetune the hyperparameters to bring down the difference in scores in train and test data set.

3. Best Model:

Best model is one which has performed well on both Test and train data set and has high accuracy and AUC score. Gradient Boosting Model seems to fit the bill where the variance in scores for test and train data is not too high as also the Model Score and AUC is more than to 90% for Class 1 that is Labour Class

Model evaluation metrics for class 0 – Labour:

Variance in Scores:

Random Forest has the perfect score of 1 for training data, however, for testing data, it has performed very poorly, once again showing a highly over fit model. We can apply other parameters on Random Forest and run another model to check if the variance in score can be decreased.

High Metrics:

Two models with high scores on most evaluation metrics are Bagging Model with other parameters and Gradient Boosting Model. However, we cannot say the same for testing data – again showing high variance in scores in train and test data. This also suggest that the model is overfitted. Once can further finetune the hyperparameters to bring down the variance in scores for train and test data set.

Best Model:

Best model is one which has performed well on both Test and train data set and has high accuracy and AUC score. Gradient Boosting Model seems to fit the bill where the variance in scores for test and train data is not too high as also the Model Score and AUC is more than to 90% for both the classes – 0 and 1.

**1.8 Based on these predictions, what are the insights**

An election exit poll is a poll of voters taken immediately after they have exited the polling stations. Voter are asked who they voted for in order to predict the result of the election.

Exit polls are also used to collect demographic data about voters and to find out why they voted as they did.

Since actual votes are cast anonymously, polling is the only way of collecting this information. Politicians mostly use public opinion data to decide what stance they should take on various political issues.

Political parties develop their election rallies and campaigns basis this information that underline certain key social, economic and cultural sentiments of the general public.

Recommendations:

Basis the objective of the business problem, we can recommend the following course of action

1. To build a new election campaign

If the primary objective of the business problem is to develop a new marketing campaign for a political party, it is essential to understand what are the sentiments of 80% of the population.

In this case, a model that predicts accurately on either class 0 or class 1 will effectively be able to help identify the sentiments of the masses.

Hence any model with higher F1 score on either class – Labour or Conservative can be considered.

2. To check which party seems more likely to win

If the exit poll is conducted to essentially predict the outcome of an ongoing election, a model with higher accuracy score should be taken into consideration.

This will help predicting results accurately 90% of the times thus depicting the true scenario in the real elections

3. To detect any fraudulent activities

Many a times election booths are rigged to produce a certain desired outcome. One may conduct exit polls across different regions or constituencies to detect fraudulent activity within that region.

If the exit poll results show a certain direction as against the actual elections, one can identify the source where the fraud has occurred – thus leading to re-elections.

In such as case, a model that shows greater than 90% scores on either class 1 or 0 on Recall or Precision can be taken into consideration.

Precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned.

4. To determine whether a particular political campaign is successful or not

A successful campaign is one which has led the general public to change their preference from one party to another.

In such a case, exit poll conducted will first determine the general sentiments of the public. Next, the campaigning activity will be run rigorously to influence the mindset and ideologies of those voters who have voted for the opposite party.

The next exit poll will be conducted to check whether those voters who had initially voted for opposite party have now voted to the concerned party.

This will help evaluate the efficacy of the campaign.

The model which gives the highest score for either class 0 or class 1 can be taken into consideration to predict whether a voter will vote for the party in question.

**Problem 2:**

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

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

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

Code Snippet to extract the three speeches:

- "
- import nltk
- nltk.download('inaugural')
- from nltk.corpus import inaugural
- inaugural.fileids()
- inaugural.raw('1941-Roosevelt.txt')
- inaugural.raw('1961-Kennedy.txt')
- inaugural.raw('1973-Nixon.txt')
- "

**Introduction:**

- NLTK will provide us with everything from splitting paragraphs to sentences, splitting words, identifying the part of speech, highlighting themes, and even helping your machine understand what the text is about.

After importing the text file, we would first count the total number of characters in each file separately. Below is the code o/p to count the char from each file.

**# Number of Characters in each file:**

```
number of characters in Roosevelt : 7571
number of characters in Kennedy : 7618
number of characters in Nixon : 9991
```

Fig.33

**# Number of words in each file:**

- Below we are counting the total number of words from each file separately.
- Here we are using the split() to split up the words based on space between each word and

- we are counting the total number of words by using the len() function.

```
number of words in Roosevelt : 1360

number of words in Kennedy : 1390

number of words in Nixon : 1819
```

Fig.33

**# Number of Sentences in each file:**

- Below we are counting the total number of sentences in each text file, by using lambda function.
- We are using pd.Dataframe to move the data as dictionary and then with lambda function.
- We are checking each sentence which ends with "." Using endswith() function output is as below.

| | Text | sentences |
|---|---|---|
| 0 | On each national day of inauguration since 178... | 67 |

| | Text | sentences |
|---|---|---|
| 0 | Vice President Johnson, Mr. Speaker, Mr. Chief... | 52 |

| | Text | sentences |
|---|---|---|
| 0 | Mr. Vice President, Mr. Speaker, Mr. Chief Jus... | 68 |

Fig.33

**2.2 Remove all the stopwords from the three speeches. Show the word count before and after the removal of stopwords. Show a sample sentence after the removal of stopwords.**

- We would use the library from nltk.corpus import stop words from nltk.tokenize import word_tokenize.
- We need these to remove all the English predefined words from each text file separately and with the help of tokenize we would separate each word and remove all the words from the text file.

Stop words which got removed from Roosevelt speech:

```
stop words which got removed {'in', 'to', 'this', 'is', 'not', 'other', 'a', 'below', 'those', 'its', 'will', 'have', 'a
t', 'into', 'here', 'should', 'itself', 'more', 'are', 'does', 'can', 'most', 'no', 'it', 'the', 'some', 'an', 'through',
'they', 'of', 'them', 'with', 'their', 'than', 'which', 'by', 'because', 'as', 'was', 'do', 'his', 'before', 'such', 'u
p', 'were', 'from', 'what', 'we', 'be', 'own', 'but', 'has', 'who', 'all', 'or', 'out', 'when', 'on', 'so', 'and', 'for',
'been', 'then', 'if', 'that', 'each', 'these', 'there', 'our'}
```

Stop words which got removed from Kennedy speech:

```
stop words which got removed {'in', 'same', 'doing', 'because', 'do', 'up', 'from', 'what', 'but', 'has', 'or', 'under', 'ou
t', 'for', 'that', 'whom', 'to', 'only', 'few', 'now', 'a', 'those', 'just', 'will', 'into', 'are', 'they', 'of', 'with', 't
heir', 'as', 'his', 'we', 'who', 'when', 'my', 'against', 'if', 'this', 'is', 'not', 'other', 'very', 'where', 'its', 'at',
'here', 'an', 'both', 'again', 'than', 'be', 'own', 'all', 'on', 'and', 'nor', 'each', 'there', 'our', 'off', 'any', 'have',
'itself', 'more', 'can', 'it', 'the', 'them', 'you', 'which', 'by', 'was', 'before', 'so', 'themselves', 'been', 'these', 'y
our'}
```

Stop words which got removed from Nixon speech:

```
stop words which got removed {'in', 'ourselves', 'am', 'no', 'because', 'do', 'such', 'from', 'what', 'but', 'has', 'or',
'out', 'down', 'for', 'that', 'to', 'only', 'now', 'a', 'those', 'why', 'just', 'will', 'should', 'are', 'they', 'of', 'w
ith', 'their', 'as', 'his', 'over', 'we', 'between', 'who', 'when', 'my', 'if', 'this', 'is', 'not', 'other', 'where', 'i
ts', 'at', 'here', 'an', 'both', 'again', 'than', 'be', 'how', 'own', 'all', 'on', 'and', 'he', 'each', 'there', 'me', 'o
ur', 'himself', 'about', 'any', 'have', 'had', 'more', 'can', 'most', 'it', 'the', 'too', 'you', 'which', 'by', 'was', 'b
efore', 'were', 'myself', 'so', 'themselves', 'been', 'these', 'your'}
```

Word count before removing stopwords:

- we are counting the total number of words by using the len() function.

```
number of words in Roosevelt : 1360

number of words in Kennedy : 1390

number of words in Nixon : 1819
```

Fig.33

Word count after removing the stopwords:

- number of words in Roosevelt: 871
- number of words in Kennedy: 904
- number of words in Nixon: 1094

sample sentence after the removal of stopwords:

Roosevelt :

```
After Removing all the stopwords from the speech ['On', 'national', 'da
y', 'inauguration', 'since', '1789', ',', 'people', 'renewed', 'sense',
'dedication', 'United', 'States', '.', 'In', 'Washington', "'s", 'day',
'task', 'people', 'create', 'weld', 'together', 'nation', '.', 'In', 'Li
ncoln', "'s", 'day', 'task', 'people', 'preserve', 'Nation', 'disruptio
n', 'within', '.', 'In', 'day', 'task', 'people', 'save', 'Nation', 'ins
titutions', 'disruption', 'without', '.', 'To', 'us', 'come', 'time',
',', 'midst', 'swift', 'happenings', ',', 'pause', 'moment', 'take', 'st
ock', '--', 'recall', 'place', 'history', ',', 'rediscover', 'may', '.',
'If', ',', 'risk', 'real', 'peril', 'inaction', '.', 'Lives', 'nations',
'determined', 'count', 'years', ',', 'lifetime', 'human', 'spirit', '.',
'The', 'life', 'man', 'three-score', 'years', 'ten', ':', 'little', ',',
'little', 'less', '.', 'The', 'life', 'nation', 'fullness', 'measure',
'live', '.', 'There', 'men', 'doubt', '.', 'There', 'men', 'believe', 'd
emocracy', ',', 'form', 'Government', 'frame', 'life', ',', 'limited',
```

Fig.88

Kennedy:

```
After Removing all the stopwords from the speech ['Vice', 'President', 'Joh
nson', ',', 'Mr.', 'Speaker', ',', 'Mr.', 'Chief', 'Justice', ',', 'Preside
nt', 'Eisenhower', ',', 'Vice', 'President', 'Nixon', ',', 'President', 'Tr
uman', ',', 'reverend', 'clergy', ',', 'fellow', 'citizens', ',', 'observ
e', 'today', 'victory', 'party', ',', 'celebration', 'freedom', '--', 'symb
olizing', 'end', ',', 'well', 'beginning', '--', 'signifying', 'renewal',
',', 'well', 'change', '.', 'For', 'I', 'sworn', 'I', 'Almighty', 'God', 's
olemn', 'oath', 'forebears', 'l', 'prescribed', 'nearly', 'century', 'thre
e', 'quarters', 'ago', '.', 'The', 'world', 'different', '.', 'For', 'man',
'holds', 'mortal', 'hands', 'power', 'abolish', 'forms', 'human', 'povert
y', 'forms', 'human', 'life', '.', 'And', 'yet', 'revolutionary', 'belief
s', 'forebears', 'fought', 'still', 'issue', 'around', 'globe', '--', 'beli
ef', 'rights', 'man', 'come', 'generosity', 'state', ',', 'hand', 'God',
'.', 'We', 'dare', 'forget', 'today', 'heirs', 'first', 'revolution', '.',
'Let', 'word', 'go', 'forth', 'time', 'place', ',', 'friend', 'foe', 'alik
e', ',', 'torch', 'passed', 'new', 'generation', 'Americans', '--', 'born',
'century', ',', 'tempered', 'war', ',', 'disciplined', 'hard', 'bitter', 'p
eace', ',', 'proud', 'ancient', 'heritage', '--', 'unwilling', 'witness',
'permit', 'slow', 'undoing', 'human', 'rights', 'Nation', 'always', 'commit
ted', ',', 'committed', 'today', 'home', 'around', 'world', '.', 'Let', 'ev
ery', 'nation', 'know', ',', 'whether', 'wishes', 'us', 'well', 'ill', ',',
'shall', 'pay', 'price', ',', 'bear', 'burden', ',', 'meet', 'hardship',
',', 'support', 'friend', ',', 'oppose', 'foe', ',', 'order', 'assure', 'su
rvival', 'success', 'liberty', '.', 'This', 'much', 'pledge', '--', '.', 'T
o', 'old', 'allies', 'whose', 'cultural', 'spiritual', 'origins', 'share',
```

Fig.89

Nixon:

```
After Removing all the stopwords from the speech ['Mr.', 'Vice', 'Presid
ent', ',', 'Mr.', 'Speaker', ',', 'Mr.', 'Chief', 'Justice', ',', 'Senat
or', 'Cook', ',', 'Mrs.', 'Eisenhower', ',', 'fellow', 'citizens', 'grea
t', 'good', 'country', 'share', 'together', ':', 'When', 'met', 'four',
'years', 'ago', ',', 'America', 'bleak', 'spirit', ',', 'depressed', 'pr
ospect', 'seemingly', 'endless', 'war', 'abroad', 'destructive', 'confli
ct', 'home', '.', 'As', 'meet', 'today', ',', 'stand', 'threshold', 'ne
w', 'era', 'peace', 'world', '.', 'The', 'central', 'question', 'us',
':', 'How', 'shall', 'use', 'peace', '?', 'Let', 'us', 'resolve', 'era',
'enter', 'postwar', 'periods', 'often', ':', 'time', 'retreat', 'isolati
on', 'leads', 'stagnation', 'home', 'invites', 'new', 'danger', 'abroa
d', '.', 'Let', 'us', 'resolve', 'become', ':', 'time', 'great', 'respon
sibilities', 'greatly', 'borne', ',', 'renew', 'spirit', 'promise', 'Ame
rica', 'enter', 'third', 'century', 'nation', '.', 'This', 'past', 'yea
r', 'saw', 'far-reaching', 'results', 'new', 'policies', 'peace', '.',
```

Fig.90

**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 stop words).**

- We have already removed the stopwatch in previous code using stop words.
- Now we are loop to look for any word and count the total number of occurrences.
- And we see from Roosevelt file we have the below words which are highly used in during the speech by president.

```
[('Nation', 12),
 ('Know', 10),
 ('Spirit', 9),
 ('Life', 9),
 ('Democracy', 9),
 ('Us', 8),
 ('People', 7),
 ('America', 7),
 ('Years', 6),
 ('Freedom', 6),
 ('Human', 5),
```

Fig.91

- Top 3 Words: Nation, Know, Spirit.
- And we see from Kennedy file we have the below words which are highly used in during the speech by president.

```
[('Let', 16),
 ('Us', 12),
 ('World', 8),
 ('Sides', 8),
 ('New', 7),
 ('Pledge', 7),
 ('Citizens', 5),
 ('Power', 5),
 ('Shall', 5),
 ('Free', 5),
 ('Nations', 5),
```

Fig.92

- Top 3 Words: let, us, world.
- And we see from Nixon file we have the below words which are highly used in during the speech by president.

```
[('Us', 26),
 ('Let', 22),
 ('America', 21),
 ('Peace', 19),
 ('World', 18),
 ('New', 15),
 ('Nation', 11),
 ('Responsibility', 11),
 ('Government', 10),
 ('Great', 9),
```

Fig.93

- Top 3 Words: Us, Let, America.

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

- Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance.
- Significant textual data points can be highlighted using a word cloud.
- Word clouds are widely used for analysing data from social network websites.
- Here we are creating the wordcloud for Roosevelt speech and we have imported the wordcloud by importing libraries.
- We are also making sure to remove all the substrings appearing the filtered data.

Word cloud for Roosevelt speech:

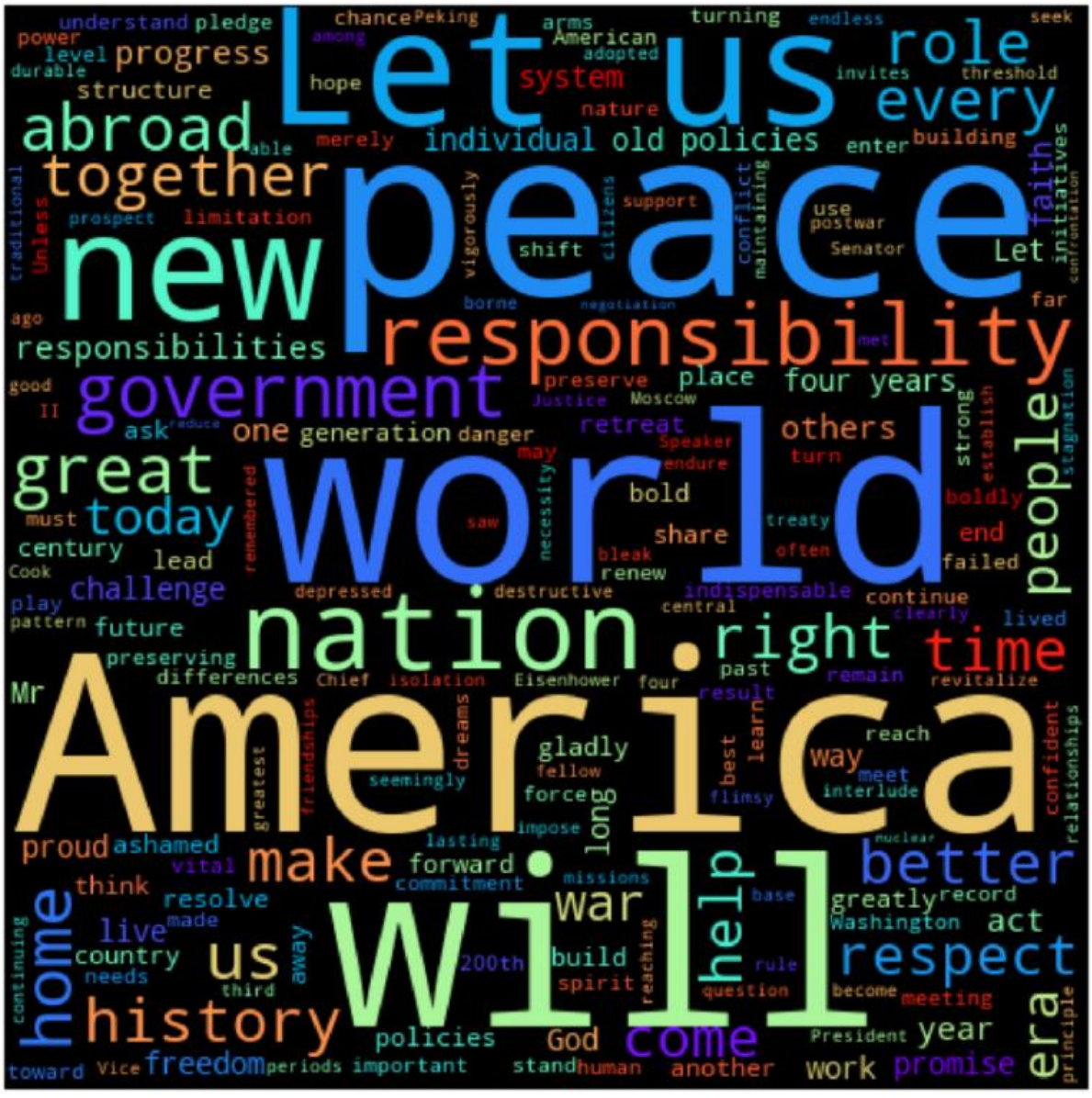


Fig.94

Word cloud for Kennedy speech:

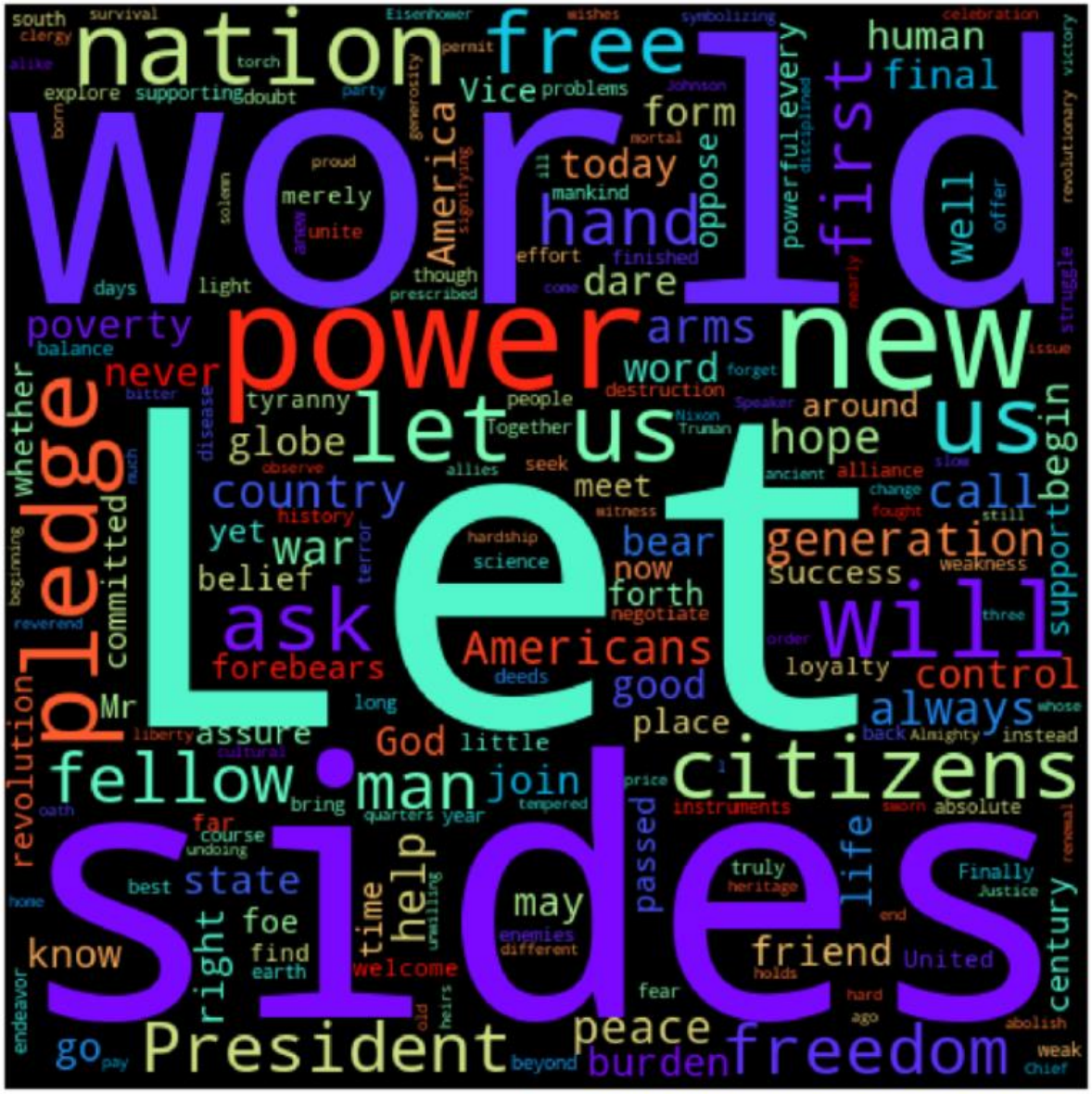


Fig.95

Word cloud for Nixon speech:

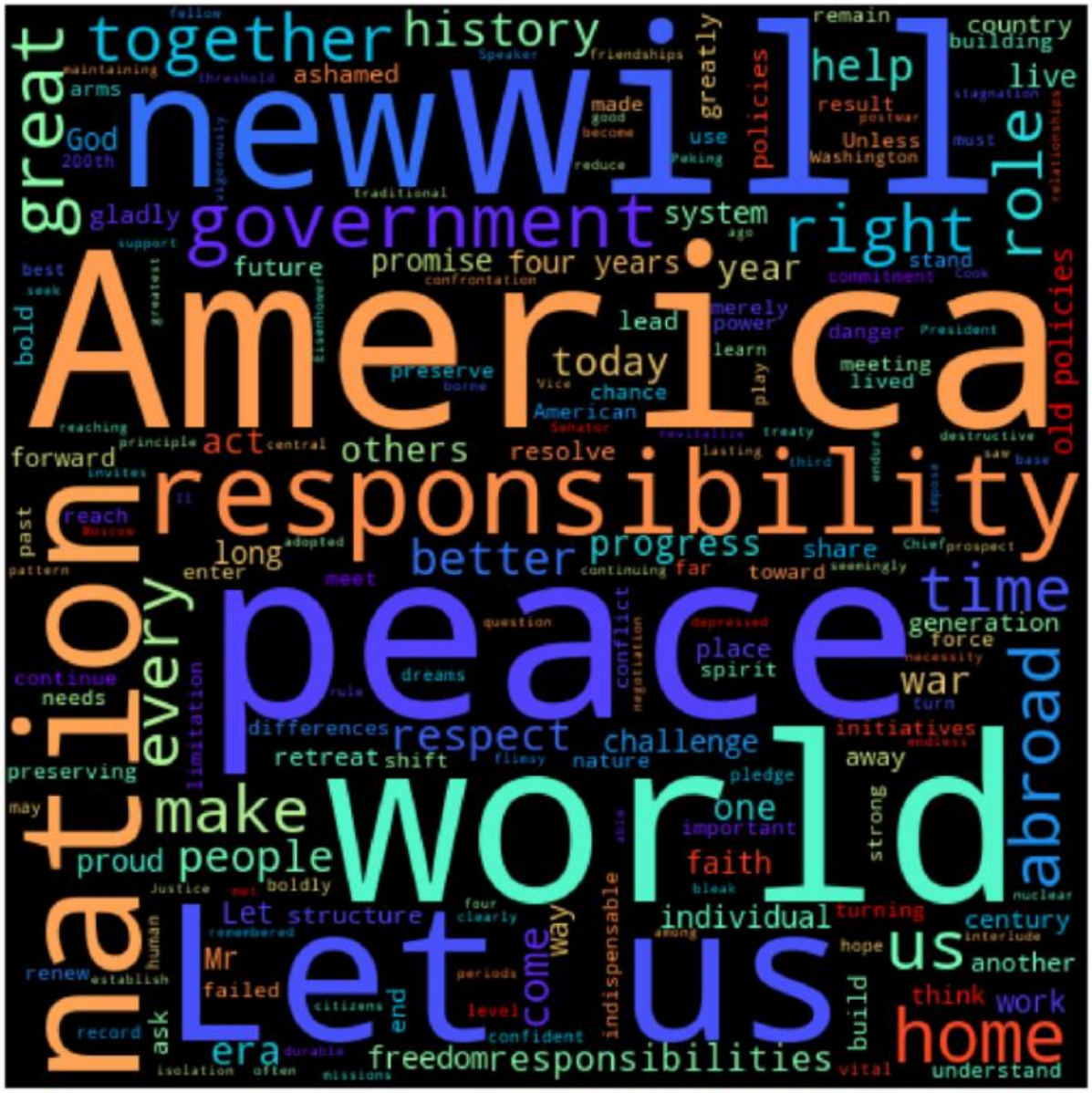


Fig.96