MACHINE LEARNING PROJECT

Submitted by,

JIYA JACOB

PGP-DSBA ONLINE

JULY-B 2021

DATE:23/01/2022

ELECTION DATA

CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

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.

INTRODUCTION

The purpose of this whole exercise is to perform various supervised learning techniques and ensemble techniques on the given data, combine all predictions and eventually find out the model with best prediction or accuracy. In supervised learning techniques, there are clearly defined X and Y variables. Supervised Learning is used to predict either a continuous response (as in regression) or a categorical response (as in classification). Ensemble Learning techniques are machine learning models for combining predictions from multiple separate models. Both regression and classification can be done using Ensemble Learning. Combining all the individual predictions can be done using either voting or averaging.

DATA DESCRIPTION

1. vote: Party choice: Conservative or Labour

2. age: in years

3. economic.cond.national: Assessment of current national economic conditions, 1 to 5.

4. economic.cond.household: Assessment of current household economic conditions, 1 to 5.

5. Blair: Assessment of the Labour leader, 1 to 5.

6. Hague: Assessment of the Conservative leader, 1 to 5.

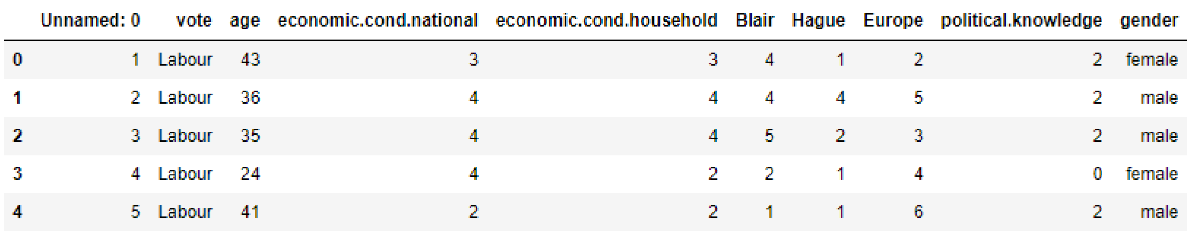
7. Europe: an 11-point scale that measures respondents' attitudes toward European integration. High scores represent ‘Eurosceptic’ sentiment.

8. political.knowledge: Knowledge of parties' positions on European integration, 0 to 3.

9. gender: female or male.

Q1.1:Read the dataset. Describe the data briefly. Interpret the inferences for each. Initial steps like head () .info (), Data Types, etc. Null value check, Summary stats, Skewness must be discussed.

Sample Dataset.

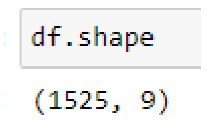


	Unnamed: 0	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	1	Labour	43	3	3	4	1	2	2	female
1	2	Labour	36	4	4	4	4	5	2	male
2	3	Labour	35	4	4	5	2	3	2	male
3	4	Labour	24	4	2	2	1	4	0	female
4	5	Labour	41	2	2	1	1	6	2	male

Table 1: Dataset Sample

The above table shows the head of the given dataset, i.e. the first five entries to ensure that the dataset has been loaded without any issues. The data consists of 1525 voters along with other attributes arranged in 9 columns such as age, gender, political knowledge etc. The first column is an index number ("Unnamed: 0"). As this only a serial no, we can remove it.

b. Shape of the dataset

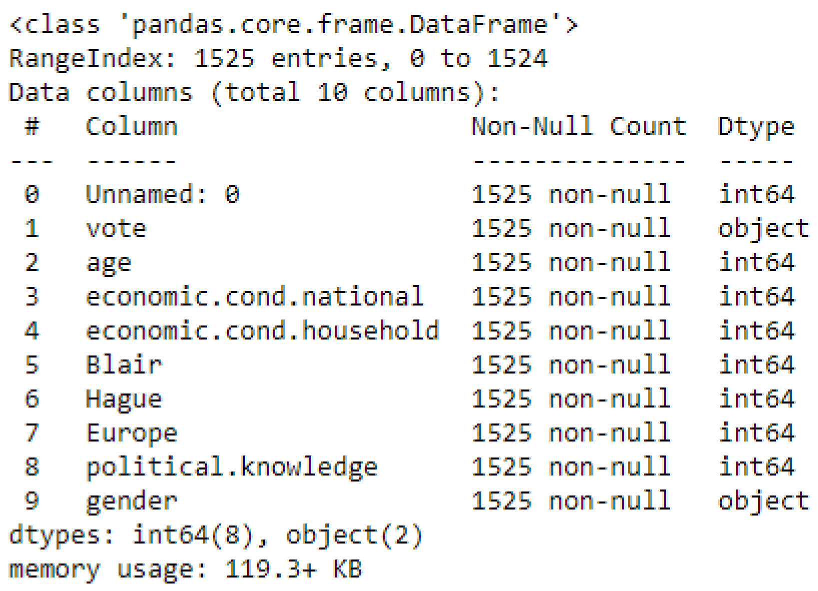


```
df.shape
```

```
(1525, 9)
```

There are total 1525 rows and 9 columns in the given dataset.

c. Checking for the missing values in the dataset.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 10 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   Unnamed: 0               1525 non-null   int64 
 1   vote                     1525 non-null   object
 2   age                      1525 non-null   int64 
 3   economic.cond.national   1525 non-null   int64 
 4   economic.cond.household  1525 non-null   int64 
 5   Blair                    1525 non-null   int64 
 6   Hague                    1525 non-null   int64 
 7   Europe                   1525 non-null   int64 
 8   political.knowledge      1525 non-null   int64 
 9   gender                   1525 non-null   object
dtypes: int64(8), object(2)
memory usage: 119.3+ KB
```

Table 2. Checking for the missing values in the dataset

From the above results we can see that there is no missing value present in the given dataset.

d. Checking for the null values in the given dataset

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

Table 3. Checking for the null values in the dataset

From the above table we can see that there is no null values present in the given dataset.

e. Checking for the datatype of variables present in the dataset.

```
Unnamed: 0                  int64
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
```

Table 4. Checking the data type of variables in the data set

From the above table we can see that out of 9 variables, the data type of the 2 variables, namely vote and gender is of object data type while the rest of the variables are of int data type.

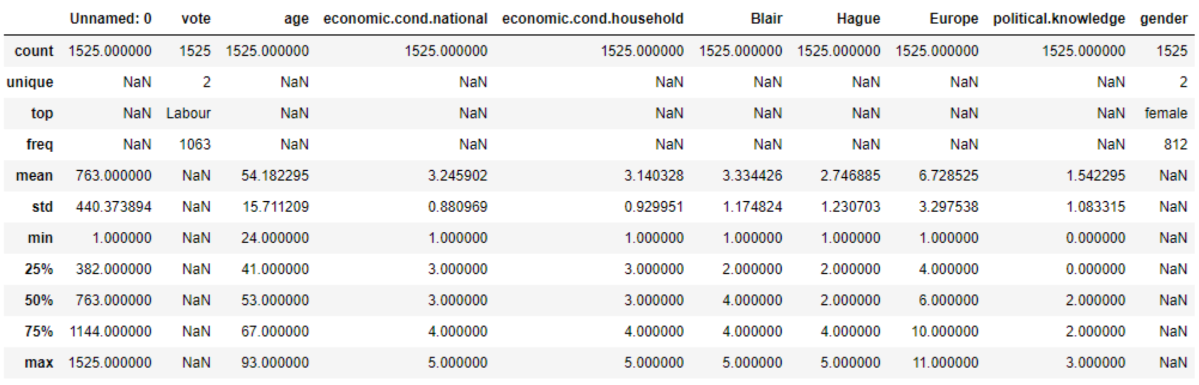
f. Checking for the number of duplicated values

```
df.duplicated().sum()
```

```
0
```

From the above output we can see that there are no duplicated values in the given dataset.

g. Checking for the summary of the given dataset.

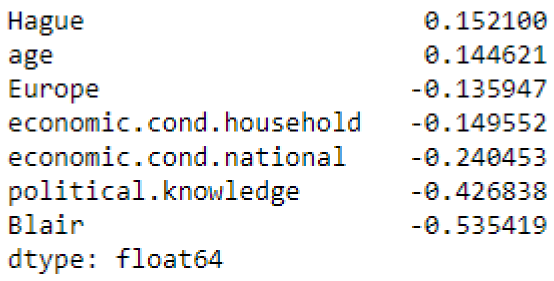


	Unnamed: 0	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
count	1525.000000	1525	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000	1525
unique	NaN	2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2
top	NaN	Labour	NaN	NaN	NaN	NaN	NaN	NaN	NaN	female
freq	NaN	1063	NaN	NaN	NaN	NaN	NaN	NaN	NaN	812
mean	763.000000	NaN	54.182295	3.245902	3.140328	3.334426	2.746885	6.728525	1.542295	NaN
std	440.373894	NaN	15.711209	0.880969	0.929951	1.174824	1.230703	3.297538	1.083315	NaN
min	1.000000	NaN	24.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	NaN
25%	382.000000	NaN	41.000000	3.000000	3.000000	2.000000	2.000000	4.000000	0.000000	NaN
50%	763.000000	NaN	53.000000	3.000000	3.000000	4.000000	2.000000	6.000000	2.000000	NaN
75%	1144.000000	NaN	67.000000	4.000000	4.000000	4.000000	4.000000	10.000000	2.000000	NaN
max	1525.000000	NaN	93.000000	5.000000	5.000000	5.000000	5.000000	11.000000	3.000000	NaN

TABLE 5: SUMMARY OF THE DATA

From the above table, we can see that we have both categorical and continuous data. For categorical data we have vote and gender and for continuous data we have age, Blair, Hague, Europe, political knowledge, economic.cond.national, economic.cond.household. vote will be target variable. From the summary of the dataset, we can see that mean of the age is the highest, followed by Europe, while the mean for political knowledge is least among all. The variables economic.cond.national and Blair share almost the similar mean whereas economic.cond.household has slightly lower mean than economic.cond.national. Similarly, age has the highest standard deviation whereas economic.cond.national has the lowest standard deviation, followed by economic.cond.household that has slightly higher standard deviation.

h. Skewness



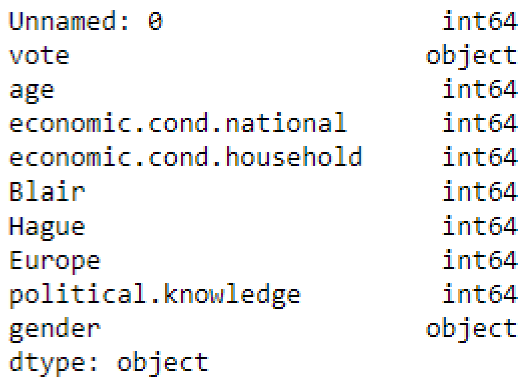
```
Hague                        0.152100
age                          0.144621
Europe                      -0.135947
economic.cond.household     -0.149552
economic.cond.national      -0.240453
political.knowledge         -0.426838
Blair                       -0.535419
dtype: float64
```

From the above given table, we can conclude that the variables ‘Hague’ and ’age’ is positively skewed while the rest of the variables are negatively skewed. Out of the negatively skewed variables, ‘Europe’ has the highest skewness followed by ‘economic.cond.household’, while ‘Blair’ has the least skewness. Among the positively skewed variables it is ‘Hague’ is slightly more positively skewed than ‘age’.

Q1.2: Perform EDA (Check the null values, Data types, shape, Univariate, bivariate analysis). Also check for outliers (4 pts). Interpret the inferences for each (3 pts) Distribution plots(histogram) or similar plots for the continuous columns. Box plots, Correlation plots. Appropriate plots for categorical variables. Inferences on each plot.

Outliers proportion should be discussed, and inferences from above used plots should be there. There is no restriction on how the learner wishes to implement this but the code should be able to represent the correct output and inferences should be logical and correct.

A. Checking for the datatype of variables present in the dataset.

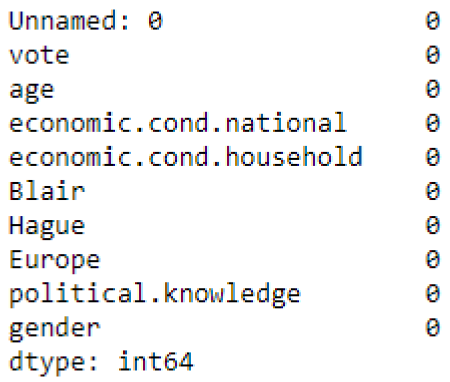


```
Unnamed: 0                  int64
vote                       object
age                         int64
economic.cond.national      int64
economic.cond.household     int64
Blair                       int64
Hague                       int64
Europe                      int64
political.knowledge     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

Checking the data type of variables in the data set

From the above table we can see that out of 9 variables, the data type of the 2 variables, namely vote and gender is of object data type while the rest of the variables are of int data type.

B. Checking for the null values in the given dataset

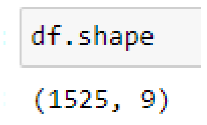


```
Unnamed: 0                 0
vote                       0
age                        0
economic.cond.national     0
economic.cond.household    0
Blair                      0
Hague                      0
Europe                     0
political.knowledge        0
gender                     0
dtype: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

Checking for the null values in the dataset

From the above table we can see that there is no null values present in the given dataset.

C. Shape of the dataset



```
df.shape
```

```
(1525, 9)
```

There are total 1525 rows and 9 columns in the given dataset.

D. Graphical Representation of univariate and bivariate analysis for continuous column

1.Age

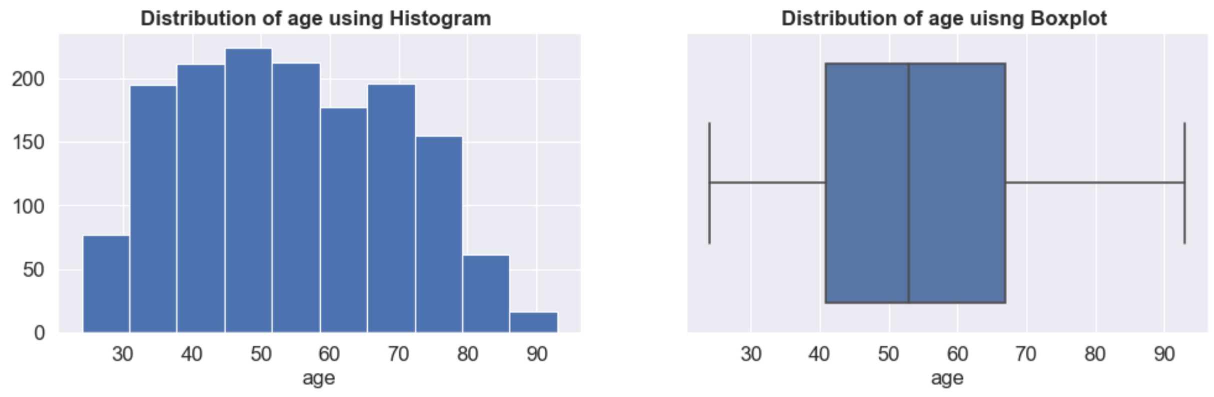


Fig 1: The box plot and the histogram showing the distribution of data of 'Age' variable

From the univariate analysis using histogram, 'age'(in years) of the voters is plotted along the x-axis. We see most the people within the age group of 45-52 constitute the largest portion of the voters (approximately 240 people),f followed by people in the age band of 38-44 and 53-59(approximately 215 people) .The voters of the age group 85-93 constitutes the least portion (approximately 35 people),after which comes the people of the age group 18-31(approximately 75 people). From the bivariate analysis using boxplot, there is no presence of outliers. The second quartile(Q2) or median for the age variable is about 53. The lower or first quartile(Q1) is about 41 and the upper or the third quartile(Q3) is about 67. The inter quartile range (IQR) for the above boxplot is 26.

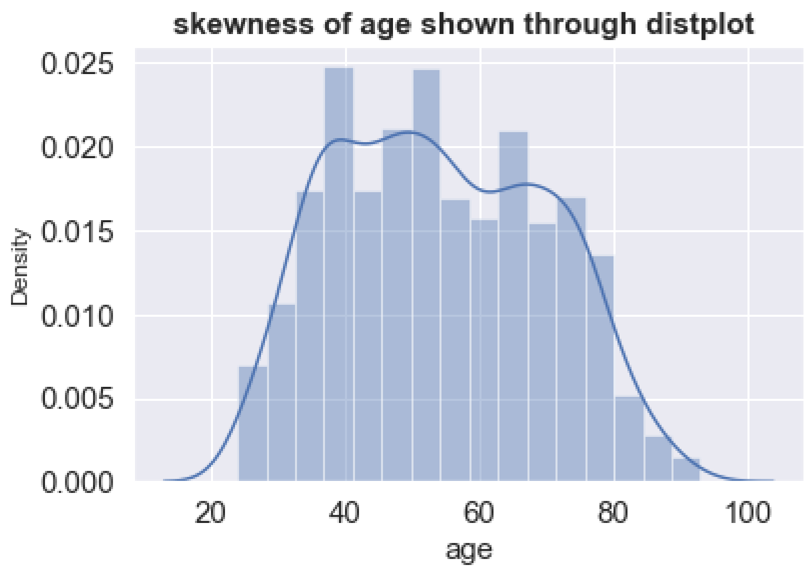


Fig 2: The distplot showing the skewness of data of 'age' variable

From the above distplot , we can see that carat is positively skewed or right skewed, with median=53,mode=0 and 37,mean= 54.18. Carat is a bi modal variable with two modes.

E. Graphical Representation of univariate and bivariate analysis for categorical columns

1. economic.cond.national

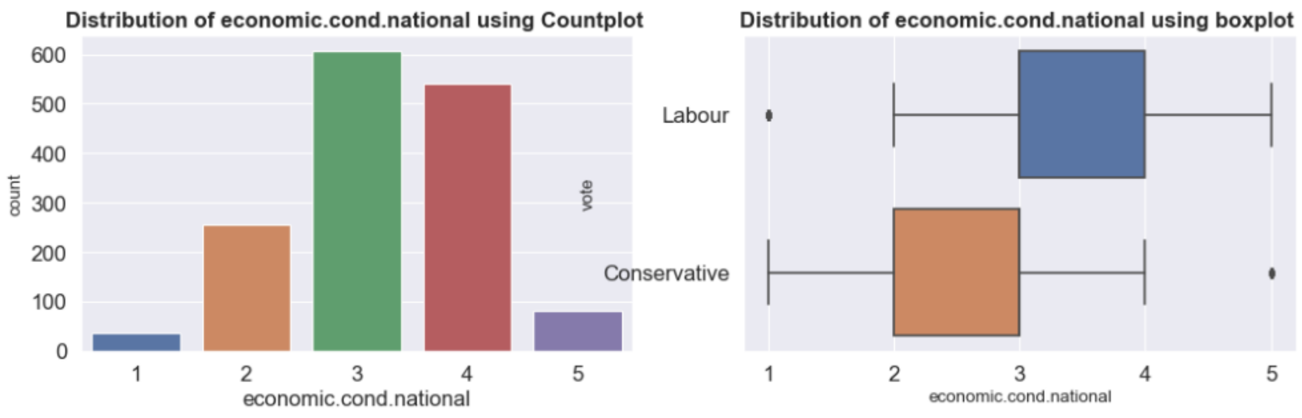


Fig 3: The box plot and the count plot showing the distribution of data of 'economic.cond.national' variable

In the univariate analysis using Count plot, assessment of current national economic conditions,1 to 5, done by the voters is plotted. In the above count plot, voters who gave a score of 3 constitute the major part ,followed by people who gave a score of 4.People who gave the score of 1 is the least ,followed by people who gave the score of 5.Futhermore,we see that 607 voters have given the current national economic condition a score of 3, assessing it to be of moderate level, followed by 542 voters giving a score of 4, through which they share the opinion of the national economic conditions to be fairly good. Only 82 people is of opinion that national economic condition is of excellent level by giving it a score of 5. 37 people have given a score of 1 and they share the thought that economic conditions are poor, while 257 voters give a score of 2 for the economic condition. From the bivariate analysis using boxplot, we can see that there is presence of outliers in both the parties.

2. Blair

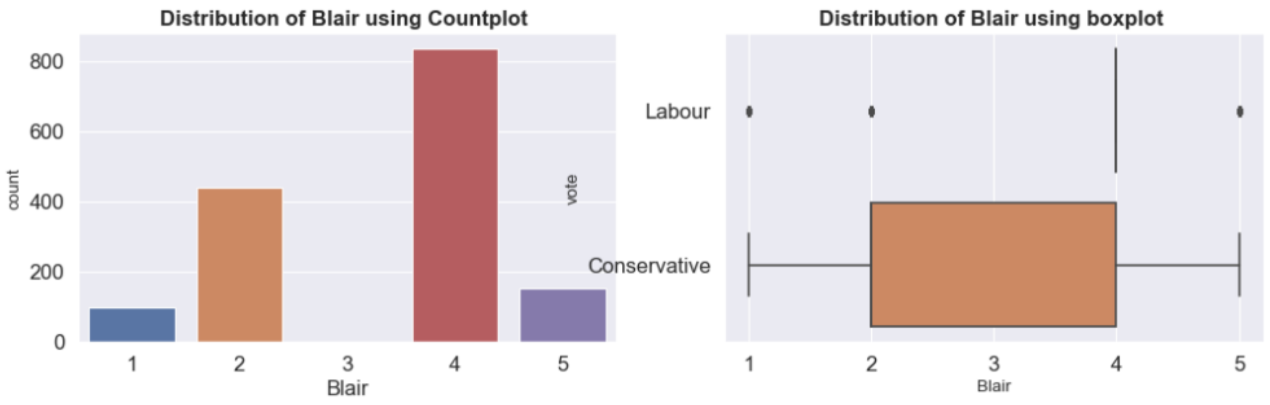


Fig 4: The box plot and the count plot showing the distribution of data of 'Blair' variable

In the univariate analysis using Count plot, assessment of the Labour leader, Tony Blair, done by the voters (1 to 5) is plotted. In the above count plot, voters who gave a score of 4 constitute the major part ,followed by people who gave a score of 2.People who gave the score of 3 is the least ,followed by people who gave the score of 5.Futhermore, we see that 836 voters have given Blair a score of 4, thereby sharing the opinion of him being a potentially good leader, followed by 438 voters giving him a score of 2, giving Blair the sign that he has a lot to improve to be perfect candidate. Only 153 people is of opinion that Blair is the ideal leader by giving him a score of 5. 97 people have given a score of 1 to Blair which indicates that he is not a perfect fit for the position. Interestingly, only one voter shares the opinion of Blair to be a good leader and thereby giving him a score of 3. From the bivariate analysis using boxplot, we can see that there is presence of outliers in both the parties.

3. economic.cond.household

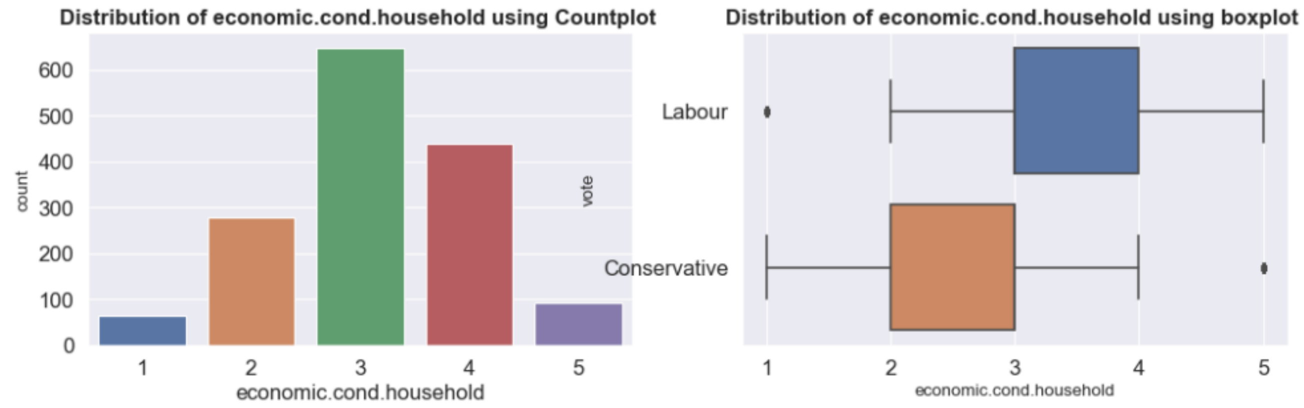


Fig 5: The box plot and the count plot showing the distribution of data of 'economic.cond.household' variable

In the univariate analysis using Count plot, assessment of current household economic conditions,1 to 5, done by the voters is plotted. In the above count plot, voters who gave a score of 3 constitute the major part ,followed by people who gave a score of 4.People who gave the score of 1 is the least ,followed by people who gave the score of 5.Futhermore,we see that

648 voters have given the current household economic condition a score of 3, assessing it to be of moderate level, followed by 440 voters giving a score of 4, through which they share the opinion of the household economic conditions to be fairly good. Only 92 people is of opinion that household economic condition is of excellent level by giving it a score of 5. 65 people have given a score of 1 and they share the thought that economic conditions are poor, while 280 voters give a score of 2 for the economic condition. From the bivariate analysis using boxplot, we can see that there is presence of outliers in both the parties.

4. Hague

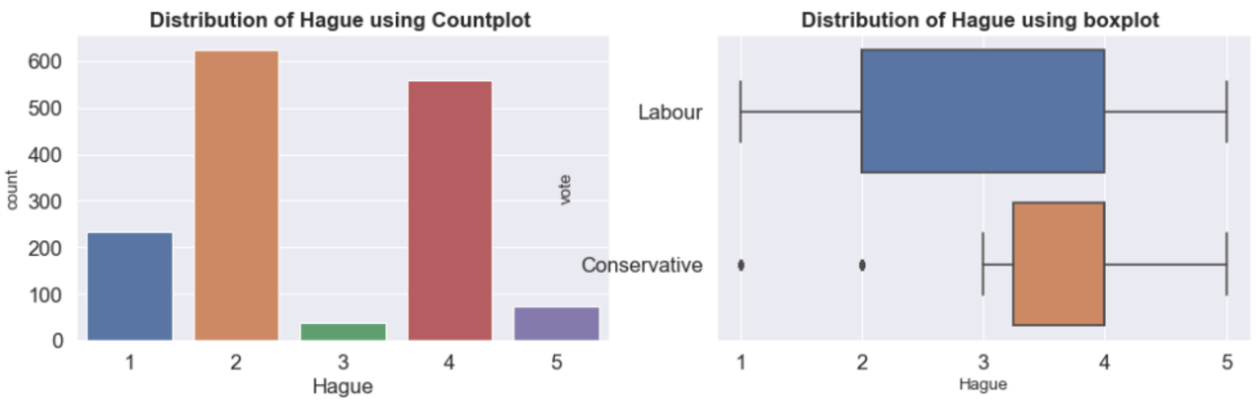


Fig 6: The box plot and the count plot showing the distribution of data of 'Hague' variable

In the univariate analysis using Count plot, assessment of the Conservative leader, Hague, done by the voters (1 to 5) is plotted. In the above count plot, voters who gave a score of 2 constitute the major part ,followed by people who gave a score of 4.People who gave the score of 3 is the least ,followed by people who gave the score of 5.Futhermore, we see that 558 voters have given Hague a score of 4, thereby sharing the opinion of him being a potentially good leader, followed by 624 voters giving him a score of 2, giving Hague the sign that he has a lot to improve to be perfect candidate. Only 73 people is of opinion that Hague is the ideal leader by giving him a score of 5. 233 people have given a score of 1 to Hague which indicates that he is not a perfect fit for the position. Interestingly, only 37 voters share the opinion of Hague to be a good leader and thereby giving him a score of 3. From the bivariate analysis using boxplot, we can see that there is presence of outliers in both the parties.

5. Europe

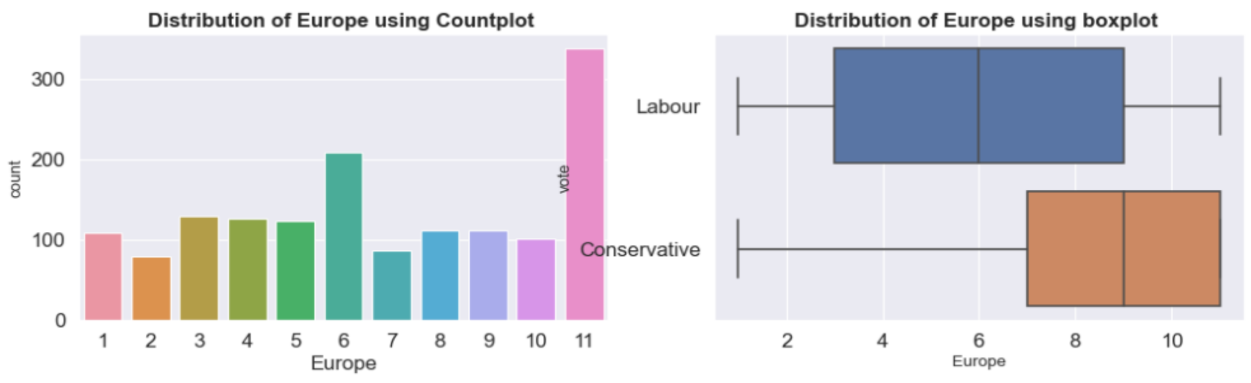


Fig 7: The box plot and the count plot showing the distribution of data of 'Europe' variable

In the univariate analysis using Count plot, an 11-point scale that measures respondents' attitudes toward European integration. High scores represent 'Eurosceptic' sentiment.is plotted. In the above count plot, voters who gave a score of 11 constitute the major part ,followed by people who gave a score of 6.People who gave the score of 2 is the least ,followed by people who gave the score of 7.Futhermore,we see that 338 voters have given the current household economic condition a score of 11, thereby expressing their 'Eurosceptic' sentiment, followed by 209 voters giving a score of 6, who shares a moderate attitude towards European integration. Only 101 people has given a score of 10, proving themselves to be close to Eurosceptic sentiment. 79 people have given a score of 2 and they have a good attitude towards European integration, while 86 voters give a score of 7 . From the bivariate analysis using boxplot, we can see that there is presence of outliers in both the parties.

6.political.knowledge

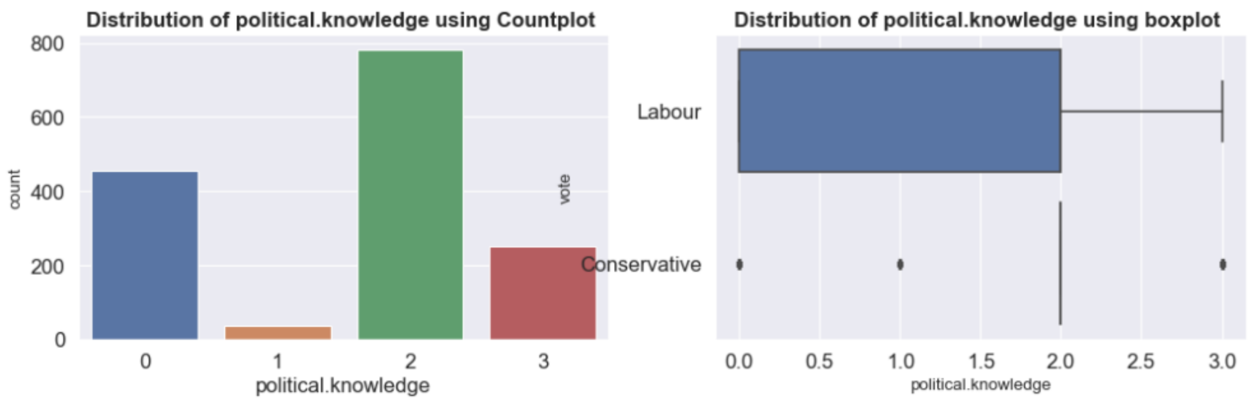


Fig 8: The box plot and the count plot showing the distribution of data of 'political.knowledge' variable

In the univariate analysis using Count plot, voter's knowledge of parties' positions on European integration, 0 to 3 is plotted. In the above count plot, voters who gave a score of 2 constitute the major part ,followed by people who gave a score of 0.People who gave the score of 1 is the least ,followed by people who gave the score of 3.Futhermore,we see that 782 voters have given their knowledge of parties' positions on European integration a score of 2 considering themselves to have a good knowledge on it, followed by 455 voters giving a score of 0, agreed that they have no knowledge on the matter. Only 250 people is of opinion that they have thorough knowledge on the parties' positions on European integration by giving their political knowledge a score of 5. 38 people have given a score of 1 and they share have some political knowledge on this issue. From the bivariate analysis using boxplot, we can see that there is presence of outliers in both the parties.

7.gender

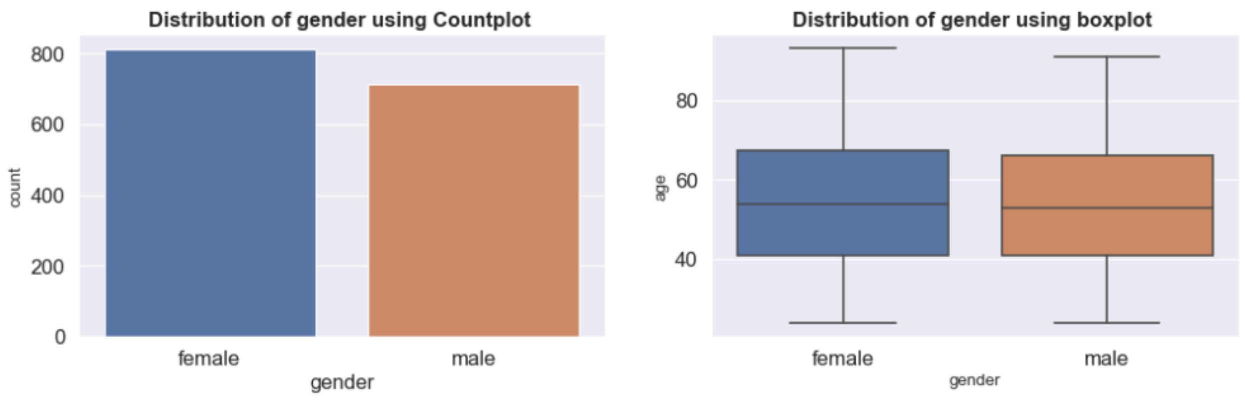


Fig 9: The box plot and the count plot showing the distribution of data of 'gender' variable

In the univariate analysis using Count plot, gender of the voters participated in the survey is plotted. From the above count plot, we can see that 812 voters are female and 713 voters are male. From the box plot we see that there is no presence of the outliers.

8.vote

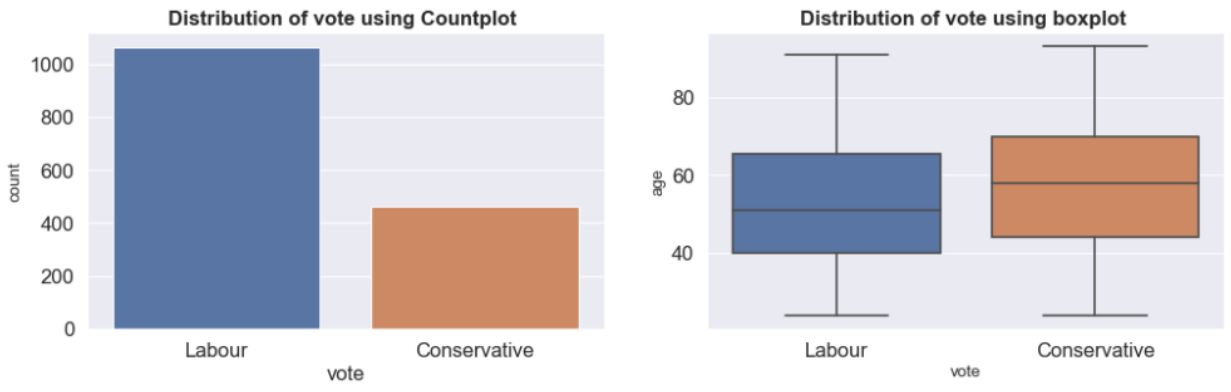


Fig 10: The box plot and the count plot showing the distribution of data of 'vote' variable

In the univariate analysis using Count plot, party choice of the voters participated in the survey is plotted. From the above count plot, we can see that 1063 voters who participated in the survey prefer Labour party, making them the majority while 462 voters, the minority portion, prefer Conservative party. From the box plot we see that there is no presence of the outliers.

F. Multivariate analysis

1.Heat Map

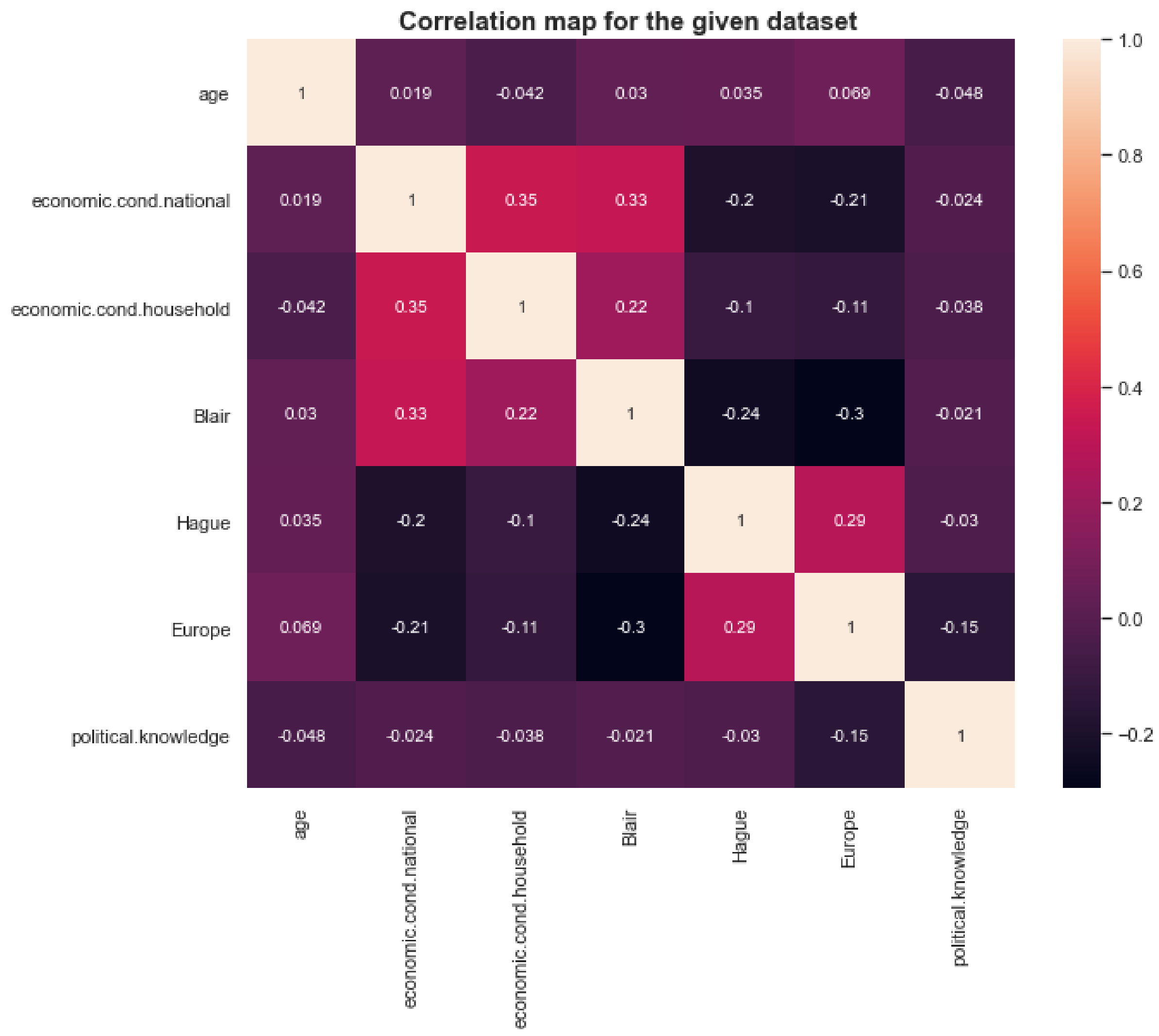
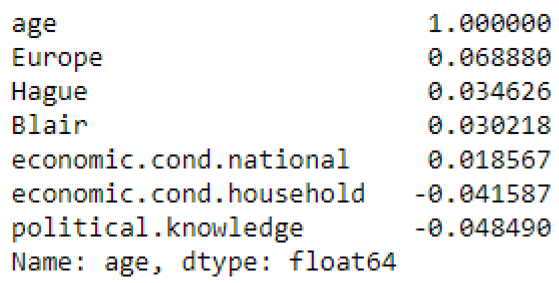


Fig 11: Heat map is portrayed for the multi variate analysis of the data.

The above heat map clearly shows the presence of multi collinearity in the dataset. It is seen that there exists positive correlation between some components, among which 'economic.cond.national' and 'economic.cond.household' has the highest positive correlation followed by 'economic.cond.national' and 'Blair'. There are many components with negative correlations among which 'Blair' and 'Europe' has the least negative correlation, while 'economic.cond.household' and 'age' is having the maximum negative correlation, followed by 'age' and 'political knowledge'.



```
age                        1.000000
Europe                     0.068880
Hague                      0.034626
Blair                      0.030218
economic.cond.national     0.018567
economic.cond.household   -0.041587
political.knowledge       -0.048490
Name: age, dtype: float64
```

TABLE 6: CORRELATION TABLE

The above table shows how each feature affects the age of the voters in CNBE dataset. It can be inferred that most of the features have very less correlation with the age of the voters, among which ‘Europe’ has the highest positive correlation, followed by ‘Hague’. The notable exceptions are "economic.cond.household" and “political knowledge” which have negative correlation (<1%).

2. Pair Plot

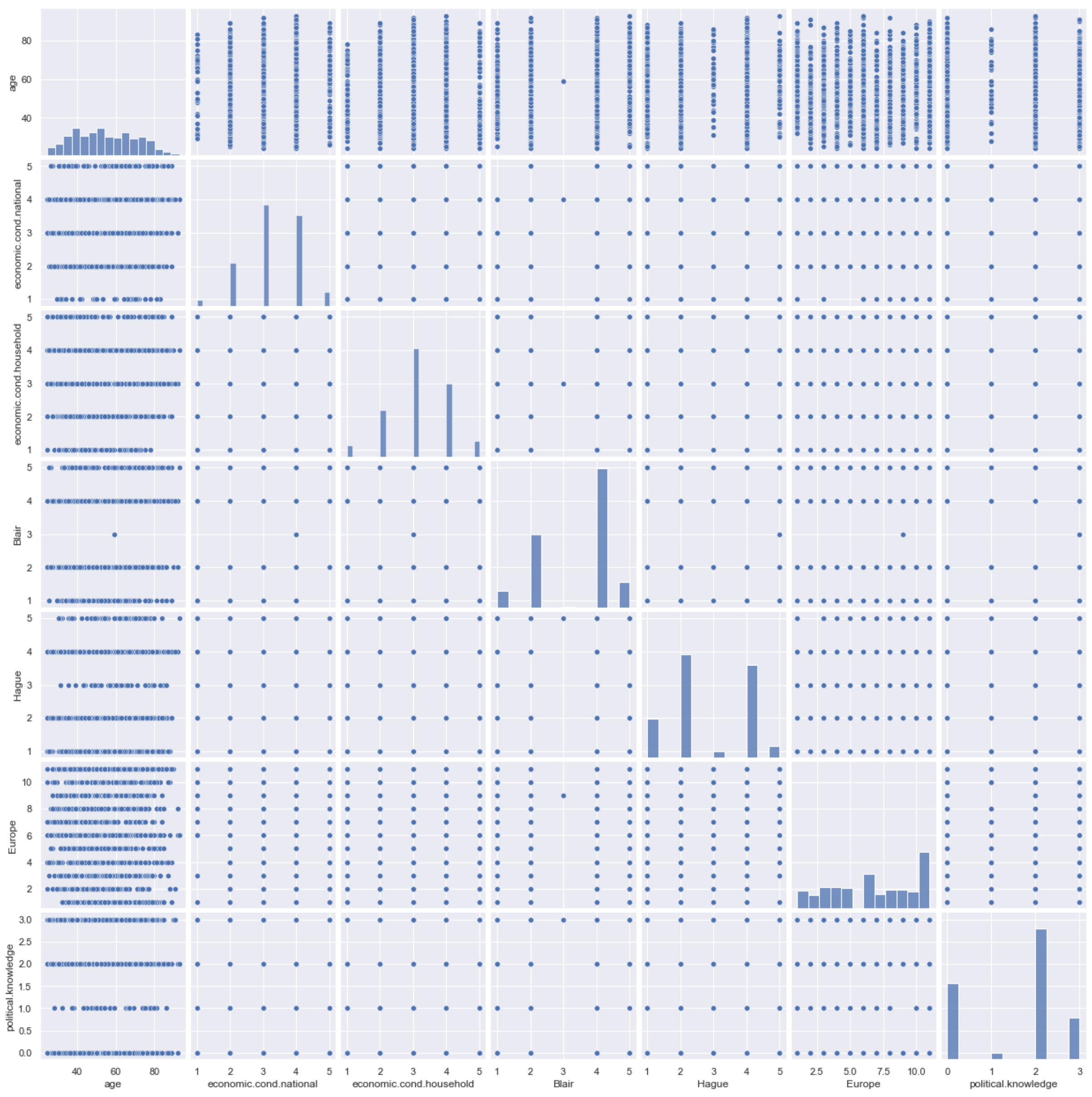


Fig 12: The pair plot showing the multivariate analysis

G. Skewness of the variables

```
Hague                      0.152100
age                        0.144621
Europe                    -0.135947
economic.cond.household   -0.149552
economic.cond.national    -0.240453
political.knowledge       -0.426838
Blair                     -0.535419
dtype: float64
```

Table 7: Skewness of all the variables

From the above given table, we can conclude that the variables ‘Hague’ and ’age’ is positively skewed while the rest of the variables are negatively skewed. Out of the negatively skewed variables, ‘Europe’ has the highest skewness followed by ‘economic.cond.household’, while ‘Blair’ has the least skewness. Among the positively skewed variables it is ‘Hague’ is slightly more positively skewed than ‘age’.

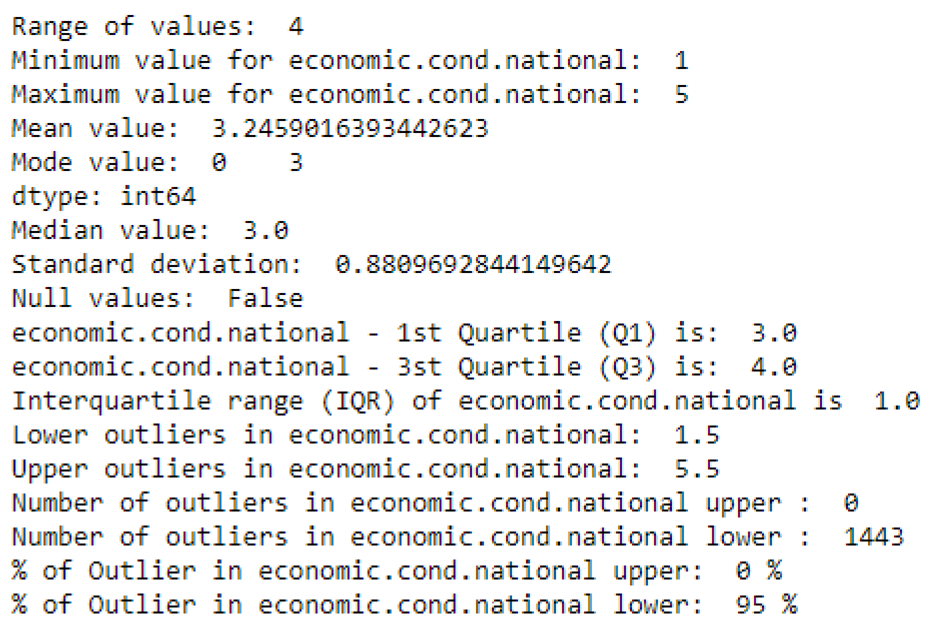
H. Outlier proportion

Outliers are not treated in this dataset since those outliers do not have any significant cause or effect on data. So those outliers do not interpret the model building also.

1.Age

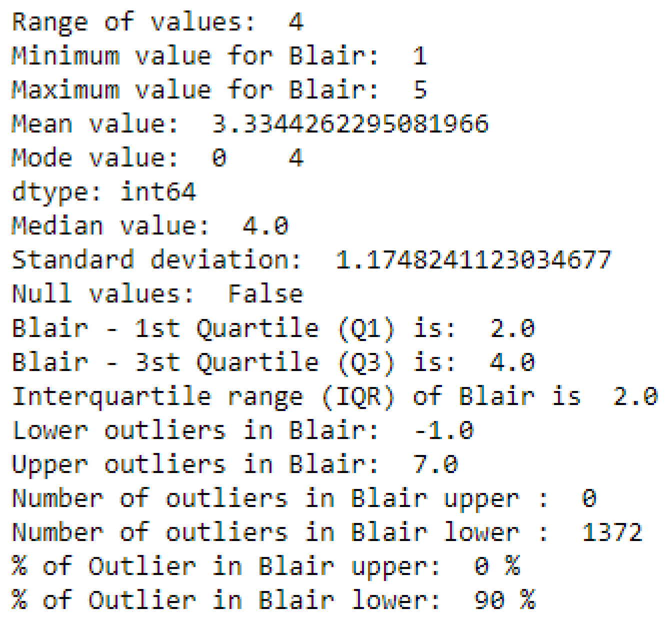
```
Range of values:  69
Minimum age:  24
Maximum age:  93
Mean value:  54.18229508196721
Mode value:  0    37
dtype: int64
Median value:  53.0
Standard deviation:  15.711208571641977
Null values:  False
age - 1st Quartile (Q1) is:  41.0
age - 3st Quartile (Q3) is:  67.0
Interquartile range (IQR) of age is  26.0
Lower outliers in age:  2.0
Upper outliers in age:  106.0
Number of outliers in age upper :  1514
Number of outliers in age lower :  0
% of Outlier in age upper:  99 %
% of Outlier in age lower:  0 %
```

2. economic.cond.national



```
Range of values:  4
Minimum value for economic.cond.national:  1
Maximum value for economic.cond.national:  5
Mean value:  3.2459016393442623
Mode value:  0    3
dtype: int64
Median value:  3.0
Standard deviation:  0.8809692844149642
Null values:  False
economic.cond.national - 1st Quartile (Q1) is:  3.0
economic.cond.national - 3st Quartile (Q3) is:  4.0
Interquartile range (IQR) of economic.cond.national is  1.0
Lower outliers in economic.cond.national:  1.5
Upper outliers in economic.cond.national:  5.5
Number of outliers in economic.cond.national upper :  0
Number of outliers in economic.cond.national lower :  1443
% of Outlier in economic.cond.national upper:  0 %
% of Outlier in economic.cond.national lower:  95 %
```

3. Blair



```
Range of values:  4
Minimum value for Blair:  1
Maximum value for Blair:  5
Mean value:  3.3344262295081966
Mode value:  0    4
dtype: int64
Median value:  4.0
Standard deviation:  1.1748241123034677
Null values:  False
Blair - 1st Quartile (Q1) is:  2.0
Blair - 3st Quartile (Q3) is:  4.0
Interquartile range (IQR) of Blair is  2.0
Lower outliers in Blair:  -1.0
Upper outliers in Blair:  7.0
Number of outliers in Blair upper :  0
Number of outliers in Blair lower :  1372
% of Outlier in Blair upper:  0 %
% of Outlier in Blair lower:  90 %
```

4. economic.cond.household

```
Range of values:  4
Minimum value for economic.cond.household:  1
Maximum value for economic.cond.household:  5
Mean value:  3.140327868852459
Mode value:  0    3
dtype: int64
Median value:  3.0
Standard deviation:  0.9299513985782148
Null values:  False
economic.cond.household - 1st Quartile (Q1) is:  3.0
economic.cond.household - 3st Quartile (Q3) is:  4.0
Interquartile range (IQR) of economic.cond.household is  1.0
Lower outliers in economic.cond.household:  1.5
Upper outliers in economic.cond.household:  5.5
Number of outliers in economic.cond.household upper :  0
Number of outliers in economic.cond.household lower :  1433
% of Outlier in economic.cond.household upper:  0 %
% of Outlier in economic.cond.household lower:  94 %
```

5.Hague

```
Range of values:  4
Minimum value for Hague:  1
Maximum value for Hague:  5
Mean value:  2.7468852459016393
Mode value:  0    2
dtype: int64
Median value:  2.0
Standard deviation:  1.2307034736168108
Null values:  False
Hague - 1st Quartile (Q1) is:  2.0
Hague - 3st Quartile (Q3) is:  4.0
Interquartile range (IQR) of Hague is  2.0
Lower outliers in Hague:  -1.0
Upper outliers in Hague:  7.0
Number of outliers in Hague upper :  0
Number of outliers in Hague lower :  1452
% of Outlier in Hague upper:  0 %
% of Outlier in Hague lower:  95 %
```

6. Europe

```
Range of values:  10
Minimum value for Europe:  1
Maximum  value for Europe:  11
Mean value:  6.728524590163935
Mode value:  0    11
dtype: int64
Median value:  6.0
Standard deviation:  3.297538370463229
Null values:  False
Europe - 1st Quartile (Q1) is:  4.0
Europe - 3st Quartile (Q3) is:  10.0
Interquartile range (IQR) of Europe is  6.0
Lower outliers in Europe:  -5.0
Upper outliers in Europe:  19.0
Number of outliers in Europe upper :  0
Number of outliers in Europelower :  444
% of Outlier in Europe upper:  0 %
% of Outlier in Europe lower:  29 %
```

7. political. knowledge

```
Range of values:  3
Minimum value for political.knowledge:  0
Maximum value for political.knowledge:  3
Mean value:  1.5422950819672132
Mode value:  0    2
dtype: int64
Median value:  2.0
Standard deviation:  1.0833147486432724
Null values:  False
political.knowledge - 1st Quartile (Q1) is:  0.0
political.knowledge - 3st Quartile (Q3) is:  2.0
Interquartile range (IQR) of political.knowledge is  2.0
Lower outliers in political.knowledge:  -3.0
Upper outliers in political.knowledge:  5.0
Number of outliers in political.knowledge upper :  0
Number of outliers in political.knowledge lower :  1372
% of Outlier in political.knowledge upper:  0 %
% of Outlier in political.knowledge lower:  90 %
```

Q1.3: Encode the data (having string values) for Modelling. Is Scaling necessary here or not? (2 pts), Data Split: Split the data into train and test (70:30) (2 pts). The learner is expected to check and comment about the difference in scale of different features on the bases of appropriate measure for example std dev, variance, etc. Should justify whether there is a necessity for scaling. Object data should be converted into categorical/numerical data to fit in the models. (pd.categorical().codes(), pd.get_dummies(drop_first=True)) Data split, ratio defined for the split, train-test split should be discussed.

a. Encoding the data (having string values)

	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

Table 8: Head of the dataset after the data encoding.

The columns namely, ’gender’ and ‘vote’ in the given dataset having the string values have been encoded using the one hot encoding technique. After data encoding, all the variables of object data types have been converted into int data types. Logistic regression model does not take categorical values so that we have encoded categorical values to integer for better results.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   age                      1525 non-null   int64
 1   economic.cond.national   1525 non-null   int64
 2   economic.cond.household  1525 non-null   int64
 3   Blair                    1525 non-null   int64
 4   Hague                    1525 non-null   int64
 5   Europe                   1525 non-null   int64
 6   political.knowledge      1525 non-null   int64
 7   vote_Labour              1525 non-null   uint8
 8   gender_male              1525 non-null   uint8
dtypes: int64(7), uint8(2)
memory usage: 86.5 KB
```

Table 9: Information about given data set after data type conversion.

b. Scaling the data

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
vote	1525	2	Labour	1063	NaN	NaN	NaN	NaN	NaN	NaN	NaN
age	1525.0	NaN	NaN	NaN	54.182295	15.711209	24.0	41.0	53.0	67.0	93.0
economic.cond.national	1525.0	NaN	NaN	NaN	3.245902	0.880969	1.0	3.0	3.0	4.0	5.0
economic.cond.household	1525.0	NaN	NaN	NaN	3.140328	0.929951	1.0	3.0	3.0	4.0	5.0
Blair	1525.0	NaN	NaN	NaN	3.334426	1.174824	1.0	2.0	4.0	4.0	5.0
Hague	1525.0	NaN	NaN	NaN	2.746885	1.230703	1.0	2.0	2.0	4.0	5.0
Europe	1525.0	NaN	NaN	NaN	6.728525	3.297538	1.0	4.0	6.0	10.0	11.0
political.knowledge	1525.0	NaN	NaN	NaN	1.542295	1.083315	0.0	0.0	2.0	2.0	3.0
gender	1525	2	female	812	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Table 10: Summary of the dataset before scaling

As we can see in the above summary, the standard deviation of variable 'age' is large compared to other variables which tend python to give more importance or weightage to that particular variable.

```
age                        246.842075
economic.cond.national       0.776107
economic.cond.household      0.864810
Blair                        1.380212
Hague                        1.514631
Europe                      10.873759
political.knowledge          1.173571
dtype: float64
```

Table 11: Table showing the variance of the continuous columns in the given dataset.

As we can see from the above table, the variance of the variables are so diverse, particularly to say 'age' has high variance meanwhile economic.cond.national has low variance. This clearly shows the imbalance in the values of the data columns and thus the need for scaling arises. Here we have scaled the data using the min-max technique. But while building LDA model the data shouldn't be scaled whereas while building KNN model the data should be scaled using z-score technique.

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	vote_Labour	gender_male
0	0.275362	0.50	0.50	0.75	0.00	0.1	0.666667	1	0
1	0.173913	0.75	0.75	0.75	0.75	0.4	0.666667	1	1
2	0.159420	0.75	0.75	1.00	0.25	0.2	0.666667	1	1
3	0.000000	0.75	0.25	0.25	0.00	0.3	0.000000	1	0
4	0.246377	0.25	0.25	0.00	0.00	0.5	0.666667	1	1

Table 12: Head of the dataset after scaling

c. Splitting the data into train and test

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

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

The data has been splitted into train and test data in the ratio 70:30 after dropping the predictor variable ‘vote_Labour’.

Q1.4: Apply Logistic Regression and LDA (Linear Discriminant Analysis) (2 pts). Interpret the inferences of both models (2 pts). Successful implementation of each model. Logical reason behind the selection of different values for the parameters involved in each model. Calculate Train and Test Accuracies for each model. Comment on the validness of models (over fitting or under fitting)

A. Logistic Regression

Logistic Regression is a supervised learning method for classification. It establishes relation between dependant class variable and independent variables using regression. The dependant variable is categorical i.e., it can take only integral values representing different classes. The probabilities describing the possible outcomes of a query point are modeled using a logistic function. The algorithm created below is a binary logistic regression model where the dependant variable, ’vote’ is binary in nature, i.e. it has only two values, either ‘1’ or ‘0’.

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

The above code snippet shows that Logistic regression model was built and the train data was fit into it. The code snippet has the parameters used for model building of Logistic Regression.

- **Solver:** solver is the algorithm to use in the optimization problem. The different types of solvers available are newton-cg, lbfgs, liblinear, sag and saga where lbfgs is the default one. To choose a solver, you might want to consider the following aspects:

➢ For small datasets, ‘liblinear’ is a good choice, whereas ‘sag’ and ‘saga’ are faster for large ones;
➢ For multiclass problems, only ‘newton-cg’, ‘sag’, ‘saga’ and ‘lbfgs’ handle multinomial loss;
➢ ‘liblinear’ is limited to one-versus-rest schemes.

Since we are dealing with multi class problem in this dataset where our target variable ‘vote’ is having two classes, we choose solver to be newton-cg.

- **max_iter:** Maximum number of iterations taken for the solvers to converge. Default is 100, but here 10000 is given for better accuracy and more learning by the algorithm.
- **penalty:** Penalized logistic regression imposes a penalty to the logistic model for having too many variables. This results in shrinking the coefficients of the less contributive variables towards zero. This is also known as regularization. The various values given for penalty parameter are none, l2, l1, elasticnet out of which l2 is the default choice.

- 'none': no penalty is added;
- 'l2': add a L2 penalty term and it is the default choice;
- 'l1': add a L1 penalty term;
- 'elasticnet': both L1 and L2 penalty terms are added.

Here we have given the penalty its default value, i.e, l2.

- **verbose:** Verbose is used to control the verbosity. Higher the verbosity, more the messages. By default, verbose is set to False (=0).

Here we have set Verbose to true (=1)

- **n_jobs:** n_jobs directly controls the number of cores on which package will attempt to run in parallel. Here n_jobs=2

The logistic regression model was built and was fit into the training dataset.

```
accuracy of train dataset: 0.8406747891283973
```

For the training data, we got accuracy as 84%, precision as 87%, recall as 91% and f1 score as 89.

```
accuracy of test dataset: 0.8231441048034934
```

For the testing data, we got accuracy as 82%, precision as 87%, recall as 89% and f1 score as 88.

As we see from the above train and test accuracies, this logistic regression model created here is a valid model as the difference between the training and testing data accuracies is less than 10%.

Furthermore, gird search CV is done as the part on model tuning on the base model and it is conducted to find the most appropriate parameters to build the model and thereby increasing the model efficiency and accuracy.

Important Feature Components

```
                    features      coef
0                        age -1.426393
1    economic.cond.national  1.350256
2  economic.cond.household  0.636408
3                      Blair  2.298532
4                      Hague -3.351914
5                     Europe -2.375334
6        political.knowledge -1.448020
7                gender_male  0.299223
```

The above list gives the significance of the feature components during the logistic regression model building. From the list we can see that, Blair is considered as the most significant features followed by economic.cond.national while ‘Hague’ is taken as the most insignificant features out of all. ‘age’, ’Europe’ and ‘political knowledge’ is also found to have very less significance in model building. economic.cond.household and gender_male also have some significance in logistic regression model building.

B. LDA

LDA uses linear combinations of independent variables to predict the class in the response variable of a given observation. LDA assumes that the independent variables are normally distributed and there is equal variance / covariance for the classes. LDA is popular, because it can be used for both classification and dimensionality reduction. LDA model uses Bayes’ Theorem to estimate probabilities

While building the LDA model, we do not scale the data, because LDA find its coefficients using the variation between the scaling, hence scaling doesn’t matter here.

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

The above code snippet shows that LDA model was built and the train data was fit into it.

```
accuracy of train dataset: 0.8369259606373008
```

For the training data, we got accuracy as 84%, precision as 87%, recall as 90% and f1 score as 88.

```
accuracy of test dataset: 0.8187772925764192
```

For the testing data, we got accuracy as 82%, precision as 87%, recall as 88% and f1 score as 87.

As we see from the above train and test accuracies, this linear discriminant analysis model is a valid model since there is no big difference between the training and testing data accuracy.

Important Feature Components

```
                    features      coef
0                        age -0.025622
1   economic.cond.national  0.349207
2  economic.cond.household  0.150347
3                      Blair  0.704420
4                      Hague -0.967188
5                     Europe -0.258563
6        political.knowledge -0.572343
7                gender_male  0.249082
```

The above list gives the significance of the feature components during the linear discriminant analysis model building. From the list we can see that, Blair is considered as the most significant features followed by economic.cond.national while ‘Hague’ is taken as the most insignificant features out of all. ‘age’, ’Europe’ and ‘political knowledge’ is also found to have very less significance in model building. economic.cond.household and gender_male also have some significance in LDA model building.

Q1.5: Apply KNN Model and Naïve Bayes Model (2pts). Interpret the inferences of each model (2 pts). Successful implementation of each model. Logical reason behind the selection of different values for the parameters involved in each model. Calculate Train and Test Accuracies for each model. Comment on the validness of models (over fitting or under fitting)

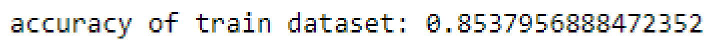
A.KNN

KNN is suited for classification where relationship between features and target classes is numerous, complex and difficult to understand and yet items in a class tend to be fairly homogenous on the values of attributes. It does not construct a “model”. Known as a non-parametric method. It can also be used for regression where the labels are continuous data and the label of query point can be average of the labels of the neighbours. KNN ‘s approach to find nearest neighbours using distance between the query point and all other points.

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

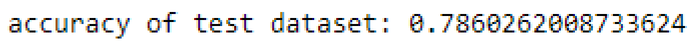
The above code snippet shows that KNN model was built and the train data was fit into it.

Before building model on KNN, the data is scaled using z-score scaling technique. We use z-score scaling to ensure all the features' distributions have mean=0 and std=1.It is useful when there are few outliers. n_neighbours is the tuning parameter of hyper parameter of knn, where it defines the number of nearest neighbours taken into consideration to predict the query point. Here, the model was built taking n_neighbours or k=5



```
accuracy of train dataset: 0.8537956888472352
```

For the training data, we got accuracy as 85%, precision as 88%, recall as 91% and f1 score as 90.



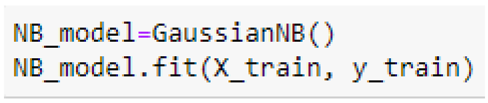
```
accuracy of test dataset: 0.7860262008733624
```

For the testing data, we got accuracy as 79%, precision as 85%, recall as 85% and f1 score as 85.

As we see from the above train and test accuracies, this knn model is overfitting where the model performs well on training data but performs poorly on testing data. This poor performance of the model is shown by the great difference between the train and test accuracies.

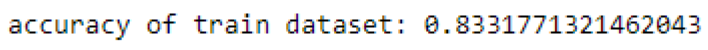
B. Naive Bayes

Naïve Bayes is one of the simplest and most effective classification algorithms which helps in building fast machine learning models that can make quick predictions. It is a probabilistic model based on Bayes' theorem, which means it predicts on the probability of an object. It is called "naive" due to the assumption that the features in the dataset are mutually independent.



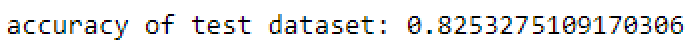
```
NB_model=GaussianNB()
NB_model.fit(X_train, y_train)
```

The above code snippet shows that Naïve Bayes model was built and the train data was fit into it.



```
accuracy of train dataset: 0.8331771321462043
```

For the training data, we got accuracy as 83%, precision as 88%, recall as 88% and f1 score as 88.



```
accuracy of test dataset: 0.8253275109170306
```

For the testing data, we got accuracy as 83%, precision as 89%, recall as 87% and f1 score as 88.

As we see from the above train and test accuracies, this Naïve Bayes model is a valid model where both the train and test accuracies are same.

Q1.6: 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.

Boosting is an ensemble meta-algorithm for primarily reducing bias, and also variance in supervised learning, and a family of machine learning algorithms that convert weak learners to strong ones. Boosting uses simple models and tries to boost their aggregate complexity. Boosting train, a large number of "weak" learners in sequence. A weak learner is a simple model that is only slightly better than random (eg. One depth decision tree). In boosting miss-classified data weights are increased for training the next model. So, training has to be done in sequence. Boosting then combines all the weak learners into a single strong learner.

A. Ada Boosting

Ada Boosting is called Adaptive Boosting. 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. In AdaBoost, the successive learners are created with a focus on the ill fitted data of the previous learner. Each successive learner focuses more and more on the harder to fit data i.e. their residuals in the previous tree

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

The above shown parameters are used for the model building. Here,

- n_estimators=100: The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early. The default value of n_estimators is 50, here we have given for more learning of the model.
- random_state =1: random_state is used to pass an int for reproducible output across multiple function calls. We are free to give any value for the random_state but ensure to give the same value everywhere.

```
accuracy of train dataset: 0.8472352389878163
```

For the training data, we got accuracy as 85%, precision as 88%, recall as 91% and f1 score as 89.

```
accuracy of test dataset: 0.8187772925764192
```

For the testing data, we got accuracy as 82%, precision as 88%, recall as 87% and f1 score as 87.

As we see from the above train and test accuracies, this Ada boosting model is a not a good model when compared to the above created models since here the difference between the model accuracies for test and train data is more than that of the above created 4 models.

B. Gradient Boosting

Gradient Boosting is another type of Boosting technique where each learner is fit on a modified version of original data (original data is replaced with the x values and residuals from previous learner). By fitting new models to the residuals, the overall learner gradually improves in areas where residuals are initially high.

```
gbcl = GradientBoostingClassifier(random_state=1)
gbcl = gbcl.fit(X_train, y_train)
```

The above code snippet shows that gradient boosting model was built and the train data was fit into it.

```
accuracy of train dataset: 0.8865979381443299
```

For the training data, we got accuracy as 89%, precision as 91%, recall as 93% and f1 score as 92.

```
accuracy of test dataset: 0.8318777292576419
```

For the testing data, we got accuracy as 83%, precision as 89%, recall as 87% and f1 score as 88.

As we see from the above train and test accuracies, this gradient boosting model is a not a good model when compared to the first 4 created models since here the difference between the model accuracies for test and train data is more than that of the first 4 models.

C. Bagging on Random Forest

Bagging is also called Booststrap Aggregation. Bagging uses complex models and tries to "smooth out" their predictions. When random subsets of the data are drawn with replacement then it is called Bagging. Bagging reduced chances of over fitting by training each model only with a randomly chosen subset of the training data. Training can be done in parallel. Bagging essentially trains a large number of "strong" learners in parallel (each model is an over fit for that subset of the data). Bagging combines (averaging or voting) these learners together to "smooth out" predictions.

```
BaggingClassifier(base_estimator=RandomForestClassifier(), n_estimators=50,
                  random_state=0)
```

The above code snippet shows that bagging model was built and the train data was fit into it. Here,

- base_estimator =RandomForestClassifier() : The base estimator to fit on random subsets of the dataset. If None, then the base estimator is decision tree classifier. Here, we have given random forest classifier as the base_estiamtor since we are doing bagging on random forest.

- n_estimator=50: The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early. The default value of n_estimators is 50.
- random_state=0: random_state is used to pass an int for reproducible output across multiple function calls. We are free to give any value for the random_state but ensure to give the same value everywhere.

```
Accuracy on Bagging train data: 0.9653233364573571
```

For the training data, we got accuracy as 97%, precision as 96%, recall as 99% and f1 score as 98.

```
Accuracy on Bagging test data: 0.8318777292576419
```

For the testing data, we got accuracy as 83%, precision as 88%, recall as 88% and f1 score as 88.

As we see from the above train and test accuracies, this bagging model is an overfitting model where there is a huge difference between train and test. This model is not a good model.

MODEL TUNING

1. LOGISTIC REGRESSION

Grid Search CV

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

Here grid search CV is conducted with the above list of parameters. We have given

1. **cv=3:** Cross validation is done to test the ability of the model to predict the new or unseen data. 3-fold cross validation is given here for checking model efficiency.
2. **max_iter=10000**: here maximum iterations is given as 10000, for better training and testing accuracies of the model.
3. **n_jobs= 2, -1:** n_jobs is the parameter that controls the number of cores on which the package will attempt to run in parallel. Here, we have given 2 and -1 which implies either the package will attempt to run in 2 cores or it will use all the cores (n_jobs= -1).
4. penalty=l2, none: here the penalty value is given as either no penalty at all (none) or the default penalty choice (l2)
5. **solver=sag, lbfgs**: here the solver is given as sag and lbfgs since this is a binary logistic regression model and only these solvers can handle multi nominal loss.

6. **tol = 0.0001, 1e-05:** here we have given very low values as the stopping criteria to ensure a greater number of iterations and more learning by the model.

```
{'penalty': 'l2', 'solver': 'sag', 'tol': 0.0001}

LogisticRegression(max_iter=10000, n_jobs=2, solver='sag')
```

The above shown is the list of the best parameters attained after Grid search CV

- **Penalty: l2**
  The default value of penalty l2 is chosen as the best penalty value by grid search CV
- **Solver: sag**
  Since the data set is having a multiclass predictor variable, sag is chosen as the best solver.
- **Tol:0.0001**
  Tolerance is the stopping criteria. Here tol is selected as 0.0001 to ensure high accuracy and more learning. More computational time is an exception as the tolerance becomes less.

- **max_iter:10000**
  Here max_iteration is set to 10000 to ensure for better accuracy and more learning by the algorithm.

- **n_jobs=2:** n_jobs is the parameter that controls the number of cores on which the package will attempt to run in parallel. Here, the n_jobs is taken as 2 which implies that it will use 2 cores of the cpu of the running system to run this package in parallel fashion.

```
accuracy of Logistic Regression train dataset after grid search CV: 0.837863167760075
```

For the training data, we got accuracy as 84%, precision as 86%, recall as 91% and f1 score as 89.

```
accuracy of Logistic Regression test dataset after grid search CV: 0.8209606986899564
```

For the testing data, we got accuracy as 82%, precision as 86%, recall as 89% and f1 score as 88.

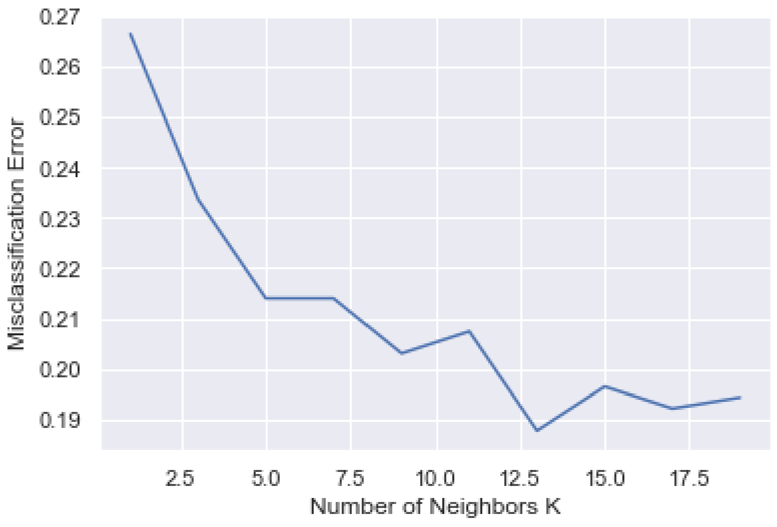
As we see from the above train and test accuracies, this logistic regression model created here is a valid model as the difference between the training and testing data accuracies is less than 10%. Interestingly, the accuracy and the performance metrics of logistic regression before and after the grid search remains the same. So, I go with the base model that was done before grid search.

2. KNN

Model Tuning

Varying the values of k

- Run the KNN with no of neighbours to be 1,3,5..19 and find the optimal number of neighbours from K=1,3,5,7....19 using the Mis classification error
- Misclassification error (MCE) = 1 - Test accuracy score. Calculated MCE for each model with neighbours = 1,3,5...19 and find the model with lowest MCE. The model with lowest MCE error is best model and the k-value corresponding to that MCE error is the ideal one.



The above figure plots the MCE error to the k-value ranging from 1 to 20. It is clear from the graph that from k=13, we get the least MCE error. It should be noted that k must be a whole number.

We get the best accuracy with k=13

```
Accuracy of the KNN train data for k=13 0.8294283036551078
```

For the training data, we got accuracy as 83%, precision as 85%, recall as 91% and f1 score as 88.

```
Accuracy of the KNN test data for k=13 0.8122270742358079
```

For the testing data, we got accuracy as 81%, precision as 86%, recall as 88% and f1 score as 87.

As we see from the above train and test accuracies, this knn model with k=13 is a valid model and has better test accuracy than the base model where k=5. Thus, we say that knn model has improved its accuracy after model was tuned, where we put k=13. So, I go with this knn model where k=13.

CROSS VALIDATION ON ALL THE MODELS

Cross validation is a technique used to assess how well our machine models perform on unseen data or test data. Cross validation tests the ability of the model to predict on new data. It is also used to flag the problems like overfitting or selection bias and give insights on how the model will generalise to an independent dataset.

Here, I have done cross validation on 4 valid models whose test accuracies are also good enough. I have done cross validation to found out which model out of these four, predicts the test data more accurately. I have 10-fold cross validation, which means the given dataset is divided randomly 10 times into train and test data and then passed to the created model.

a. Logistic regression

```
array([0.78504673, 0.82242991, 0.86915888, 0.85046729, 0.8317757 ,
       0.82242991, 0.78504673, 0.9245283 , 0.83018868, 0.80188679])
```

Train accuracies of Logistic regression model after cross validation

```
array([0.82608696, 0.86956522, 0.80434783, 0.7826087 , 0.82608696,
       0.80434783, 0.84782609, 0.89130435, 0.88888889, 0.75555556])
```

Test accuracies of LDA model after cross validation

After 10-fold cross validation, scores both on train and test data set respectively for all 10 folds are not same. Hence this model is not valid.

LDA

```
array([0.78504673, 0.82242991, 0.85981308, 0.8317757 , 0.8411215 ,
       0.82242991, 0.80373832, 0.9245283 , 0.82075472, 0.80188679])
```

Train accuracies of LDA model after cross validation

```
array([0.82608696, 0.84782609, 0.80434783, 0.80434783, 0.82608696,
       0.80434783, 0.84782609, 0.84782609, 0.91111111, 0.75555556])
```

Test accuracies of LDA model after cross validation

After 10-fold cross validation, scores both on train and test data set respectively for all 10 folds are not same. Hence this model is not valid.

b. KNN

```
array([0.79439252, 0.76635514, 0.78504673, 0.81308411, 0.81308411,
       0.77570093, 0.73831776, 0.83018868, 0.8490566 , 0.80188679])
```

Train accuracies of KNN model after cross validation

```
array([0.76086957, 0.80434783, 0.73913043, 0.7826087 , 0.82608696,
       0.76086957, 0.80434783, 0.76086957, 0.84444444, 0.73333333])
```

Test accuracies of KNN model after cross validation

After 10-fold cross validation, scores both on train and test data set respectively for all 10 folds are not same. Hence this model is not valid.

c. Naïve Bayes

```
array([0.81308411, 0.8317757 , 0.82242991, 0.85046729, 0.82242991,
       0.81308411, 0.81308411, 0.88679245, 0.82075472, 0.81132075])
```

Train accuracies of naïve Bayes model after cross validation

```
array([0.82608696, 0.84782609, 0.82608696, 0.80434783, 0.76086957,
       0.80434783, 0.84782609, 0.91304348, 0.88888889, 0.82222222])
```

Test accuracies of Naïve Bayes model after cross validation

After 10-fold cross validation, scores both on train and test data set respectively for all 10 folds are almost same. Hence this model is valid.

Thus, we can conclude from the above results that during cross validation on 4 models, in Naïve Bayes, test accuracies were greater than train accuracies 7 times, greater in number than any other model. Hence, we conclude that Naïve Bayes is the best model among all the models created.

Q1.7: 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. (3 pts)

A. Logistic Regression

1. Accuracy

```
accuracy of train dataset: 0.8406747891283973
```

```
accuracy of test dataset: 0.8231441048034934
```

Comparing both the train and test accuracies we can say that this model is a good model

2. Confusion Matrix

➢ Training Data

```
Confusion matrix for the train data
 [[230 102]
 [ 68 667]]
```

Given above is the confusion matrix for training data using logistic regression model. Here,

- True positive=230, which is actually True (value or positive=1) and has been predicted true too.
- False negative= 102, which is actually True (value or positive=1), but predicted False (value=negative or 0).

- False positive=68, which is actually False (value= negative or 0), but predicted True (1).
- True negative=667, which is actually False and has been predicted False too.

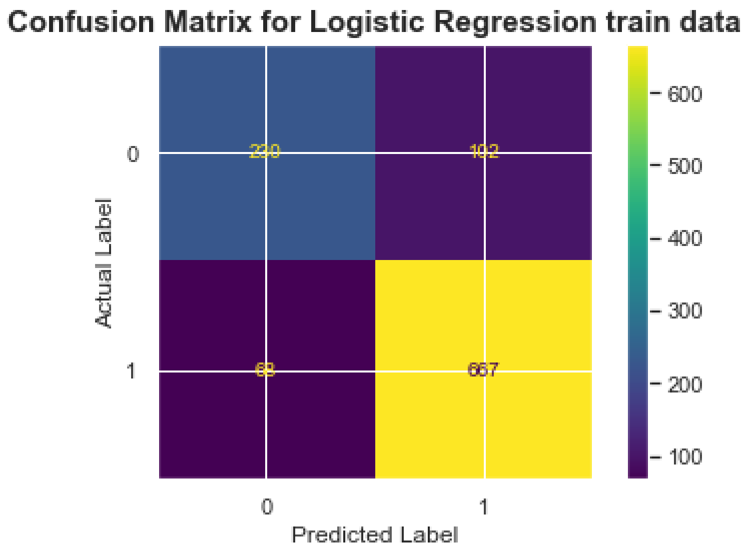


Fig 13: Confusion matrix for the Logistic regression train data

➢ Testing Data

```
Confusion matrix for the test data
 [[ 85  45]
 [ 36 292]]
```

Given above is the confusion matrix for testing data using logistic regression model. Here,

- True positive=85, which is actually True (value or positive=1) and has been predicted true too.
- False negative= 45, which is actually True (value or positive=1), but predicted False (value=negative or 0).
- False positive=36, which is actually False (value= negative or 0), but predicted True (1).
- True negative=292, which is actually False and has been predicted False too.

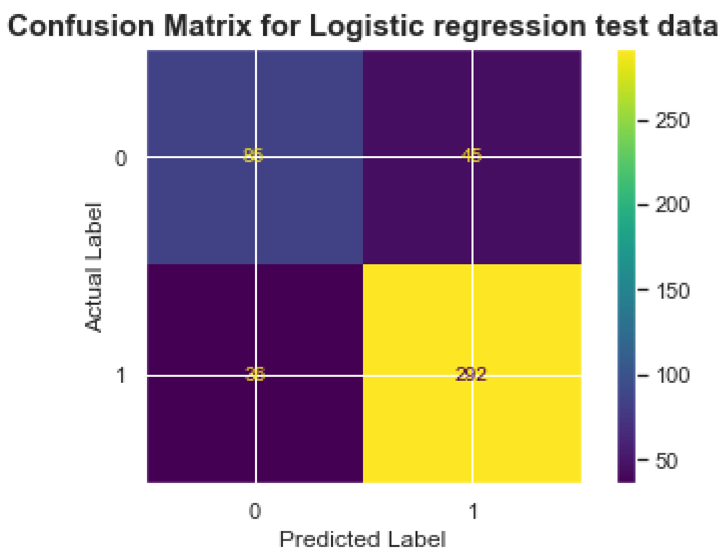


Fig 14: Confusion matrix for the Logistic regression test data

3. ROC curve

The ROC curve shows the trade-off between sensitivity (or TPR) and specificity (1-FPR). Models that gives curve close to top-left corner indicates a better performance. The closer the curve comes to 45-degree diagonal of the ROC space, the less accurate the test is. A high area under the curve represents both high recall and precision.

➢ Training Data

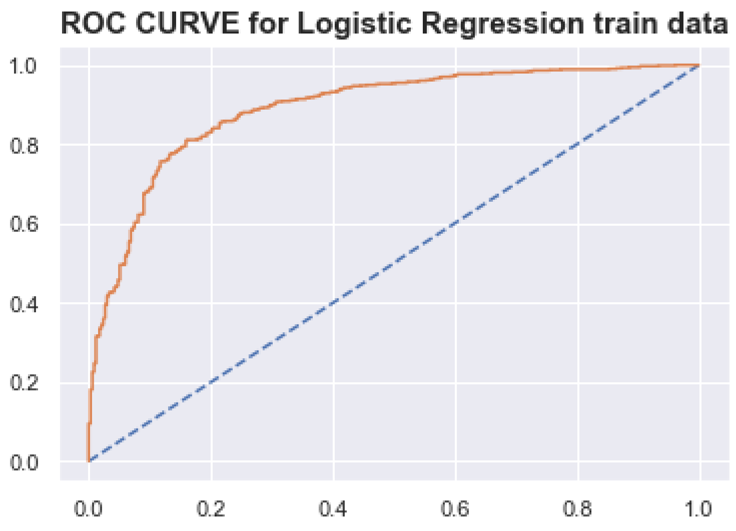


Fig 15: The ROC curve for the training data using the Logistic regression model

➢ Testing Data

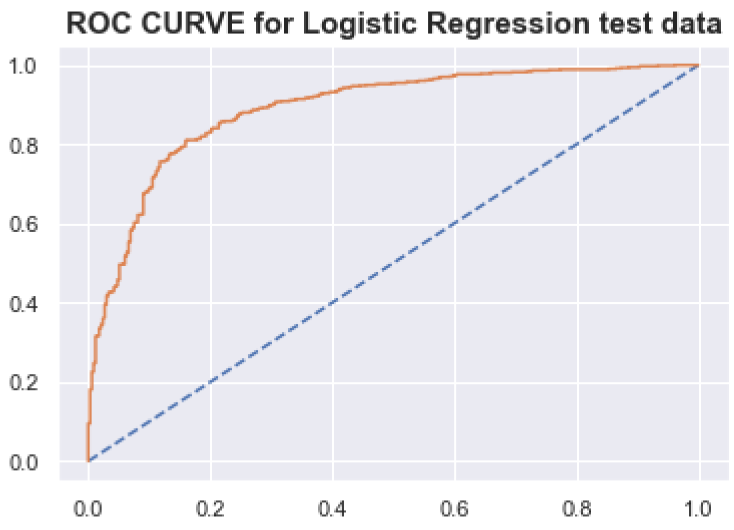


Fig 16: The ROC curve for the testing data using the Logistic regression model

4. ROC_AUC score

The ROC_AUC score is the measure of the ability of a classifier to distinguish between classes and is used as the summary of ROC curve. Higher the ROC_AUC score, the better the performance of the model at distinguishing between the positive and negative classes. To compare the models, we use ROC AUC that gives us the optimal combination of all the performance metrics.

- Training Data

```
ROC_AUC score: 0.889
```

- Testing Data

```
ROC_AUC score: 0.882
```

From the above ROC_AUC score and ROC curve, we see that this logistic regression model is a good model having a good ROC_AUC score and steady ROC curve in both train and test data.

5. Classification Report

Precision and recall are the very useful measure of success of prediction when the classes are very imbalanced.

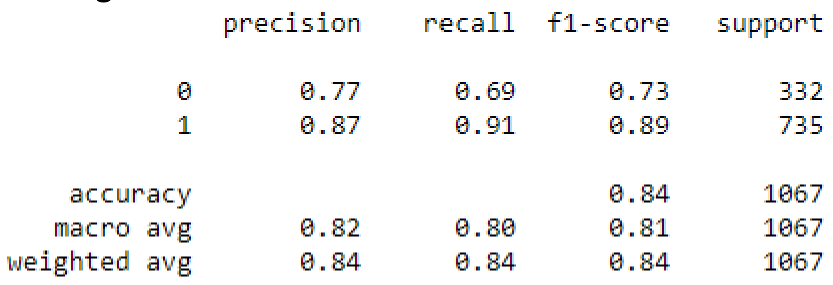
a. Sensitivity / Recall – How many of the actual True data points are identified as True data points by the model. Remember, False Negatives are those data points which should have been identified as True.
Recall=true positive/ true positive+ false negative.

b. Precision- Among the points identified as Positive by the model, how many are really positive. Precision =true positive/ true positive+ false positive.

Recall and precision will oppose each other. We want recall to be as close to 1 as possible without precision being too bad. A high precision rate refers to low false positive rate and high recall relates to a low false negative rate. Meanwhile, precision is more important than recall, means getting a false positive is very costly when compared to getting false negative.

c. **F1 score**- It is the measure of model's accuracy on a dataset. F1 score is the defined as the harmonic mean of a model's precision and recall. F1 score of 1 is considered the best and 0 the worst. A low F1 score is an indication of both poor precision and recall.

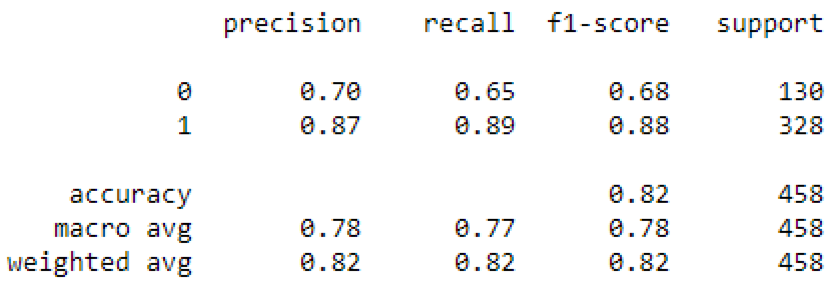
- Training Data



```
              precision    recall  f1-score   support

           0       0.77      0.69      0.73       332
           1       0.87      0.91      0.89       735

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

From the above classification report, we can see that using the logistic regression model, the precision for training data is 0.87, recall is 0.91, f1 score is 0.89 and accuracy is 0.84. A high F1 score is due to high precision and recall scores. A high F1 score also tells that this model was well trained.

- Testing Data

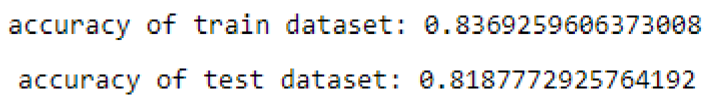


```
              precision    recall  f1-score   support

           0       0.70      0.65      0.68       130
           1       0.87      0.89      0.88       328

    accuracy                           0.82       458
   macro avg       0.78      0.77      0.78       458
weighted avg       0.82      0.82      0.82       458
```

From the above classification report, we can see that using the logistic regression model, the precision for testing data is 0.87, recall is 0.89, f1 score is 0.88 and accuracy is 0.82. Though the accuracy and f1 score is good enough, high recall score is an issue here. But still this model is considered a valid model.

B. LDA

1. Accuracy



```
accuracy of train dataset: 0.8369259606373008

 accuracy of test dataset: 0.8187772925764192
```

Comparing both the train and test accuracies we can say that this model is a good model.

2. Confusion Matrix

- Training Data

```
Confusion matrix for the train data
 [[233  99]
 [ 75 660]]
```

Given above is the confusion matrix for training data using lda model. Here,

- True positive=233, which is actually True (value or positive=1) and has been predicted true too.
- False negative= 99, which is actually True (value or positive=1), but predicted False (value=negative or 0).
- False positive=75, which is actually False (value= negative or 0), but predicted True (1).
- True negative=660, which is actually False and has been predicted False too.

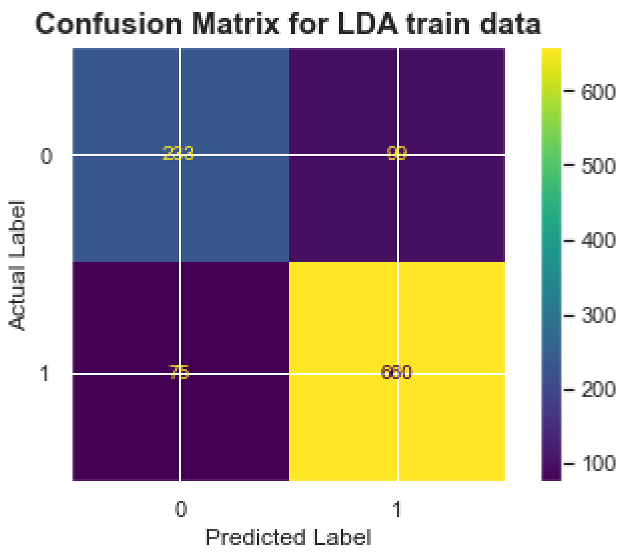


Fig 17: Confusion matrix for the LDA train data

➢ Testing Data

```
Confusion matrix for the test data
 [[ 86  44]
 [ 39 289]]
```

Given above is the confusion matrix for testing data using lda model. Here,

- True positive=86, which is actually True (value or positive=1) and has been predicted true too.
- False negative= 44, which is actually True (value or positive=1), but predicted False (value=negative or 0).
- False positive=39, which is actually False (value= negative or 0), but predicted True (1).
- True negative=289, which is actually False and has been predicted False too.

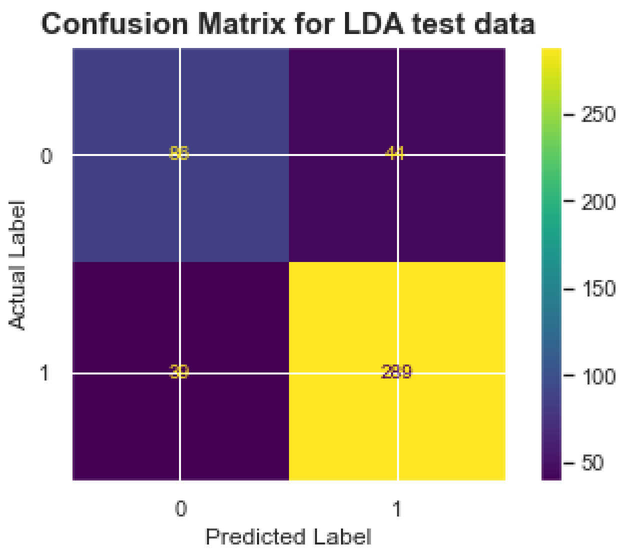


Fig 18: Confusion matrix for the LDA test data

3. ROC curve
   - Training Data

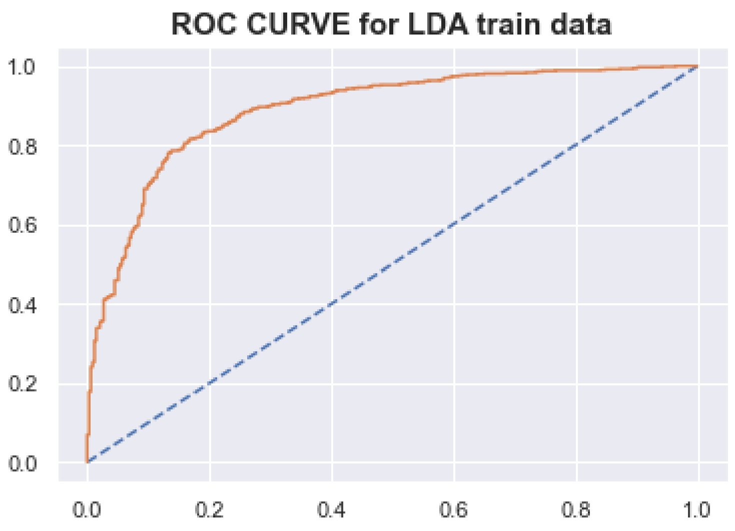


Fig 19: The ROC curve for the training data using the LDA model

   - Testing Data

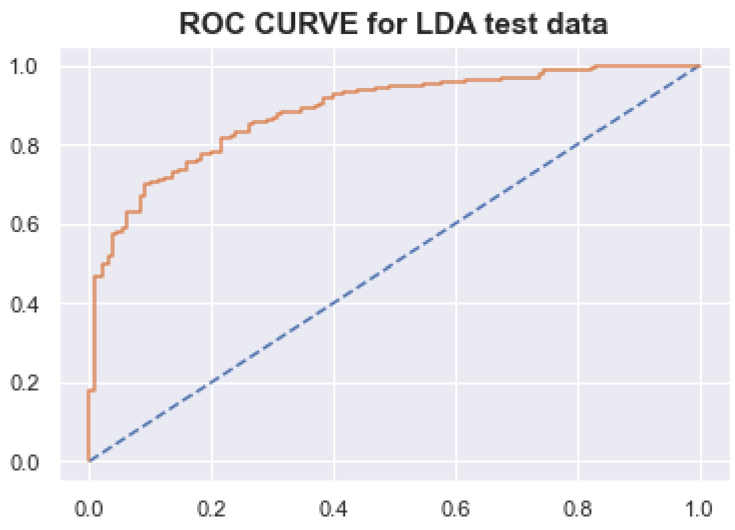


Fig 20: The ROC curve for the testing data using the LDA model

4. ROC_AUC score
   - Training Data

```
ROC_AUC score : 0.889
```

   - Testing Data

```
ROC_AUC score: 0.884
```

From the above ROC_AUC score and ROC curve, we see that this LDA model is a good model having a good ROC_AUC score and almost similar ROC curve (neglecting the minute noises in the curve) in both train and test data.

5. Classification Report
   - Training Data

```
              precision    recall  f1-score   support

           0       0.76      0.70      0.73       332
           1       0.87      0.90      0.88       735

    accuracy                           0.84      1067
   macro avg       0.81      0.80      0.81      1067
weighted avg       0.83      0.84      0.84      1067
```

From the above classification report, we can see that using the lda model, the precision for training data is 0.87, recall is 0.90, f1 score is 0.88 and accuracy is 0.84. A high F1 score is due to high precision and recall scores. A high F1 score also tells that this model was well trained.

   - Testing Data

```
              precision    recall  f1-score   support

           0       0.69      0.66      0.67       130
           1       0.87      0.88      0.87       328

    accuracy                           0.82       458
   macro avg       0.78      0.77      0.77       458
weighted avg       0.82      0.82      0.82       458
```

From the above classification report, we can see that using the lda model, the precision for testing data is 0.87, recall is 0.88, f1 score is 0.87 and accuracy is 0.82. Though the accuracy and f1 score is good enough, high recall score is an issue here. But still this model is considered a valid model.

C. Naïve Bayes

1. Accuracy

```
accuracy of train dataset: 0.8331771321462043
```

```
accuracy of test dataset: 0.8253275109170306
```

Comparing both the train and test accuracies we can say that this model is a good model.

2. Confusion Matrix

➢ Training Data

```
Confusion matrix for the train data
 [[240  92]
 [ 86 649]]
```

Given above is the confusion matrix for training data using Naïve Bayes model. Here,

- True positive=240, which is actually True (value or positive=1) and has been predicted true too.
- False negative= 92, which is actually True (value or positive=1), but predicted False (value=negative or 0).
- False positive=86, which is actually False (value= negative or 0), but predicted True (1).
- True negative=649, which is actually False and has been predicted False too.

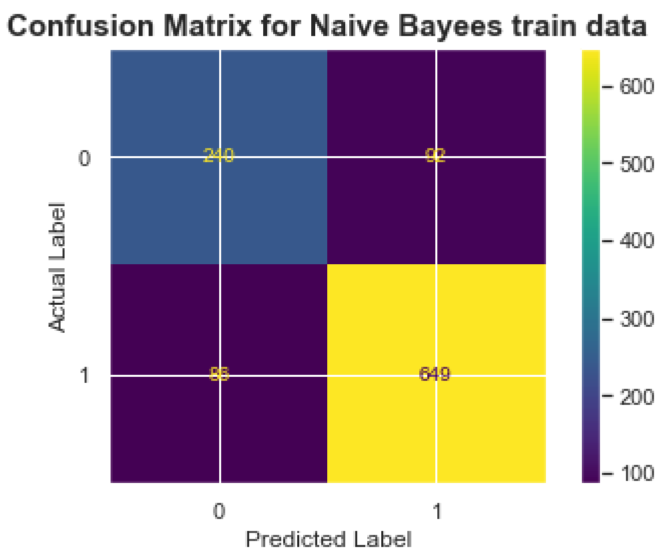


Fig 21: Confusion matrix for the Naïve Bayes train data

- ➢ Testing Data

```
Confusion matrix for the test data
 [[ 94  36]
 [ 44 284]]
```

Given above is the confusion matrix for testing data using Naïve Bayes model. Here,

- True positive=94, which is actually True (value or positive=1) and has been predicted true too.
- False negative= 36, which is actually True (value or positive=1), but predicted False (value=negative or 0).
- False positive=44, which is actually False (value= negative or 0), but predicted True (1).
- True negative=284, which is actually False and has been predicted False too.

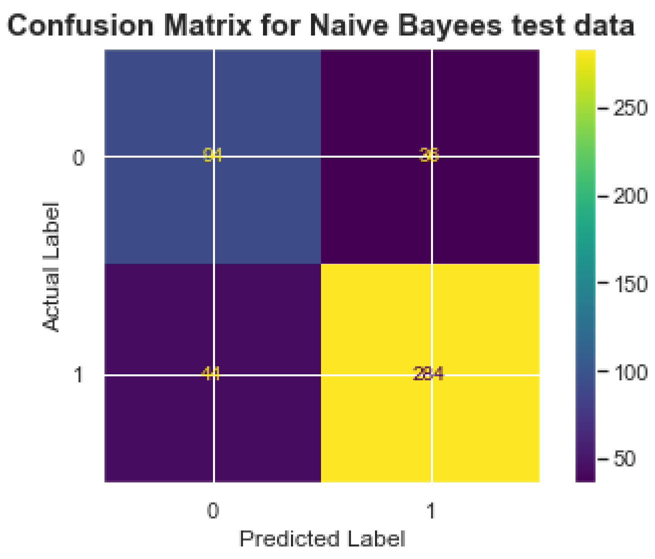


Fig 22: Confusion matrix for the Naïve Bayes test data

3. ROC curve
   - Training Data

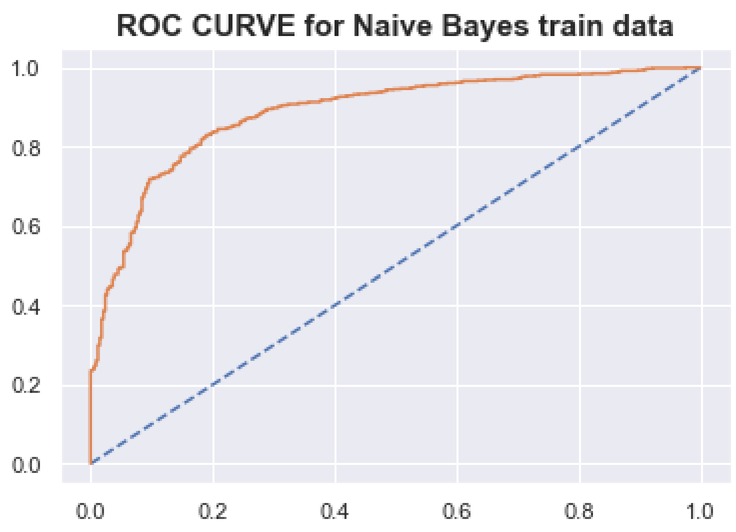


Fig 23: The ROC curve for the training data using the Naïve Bayes model

   - Testing Data

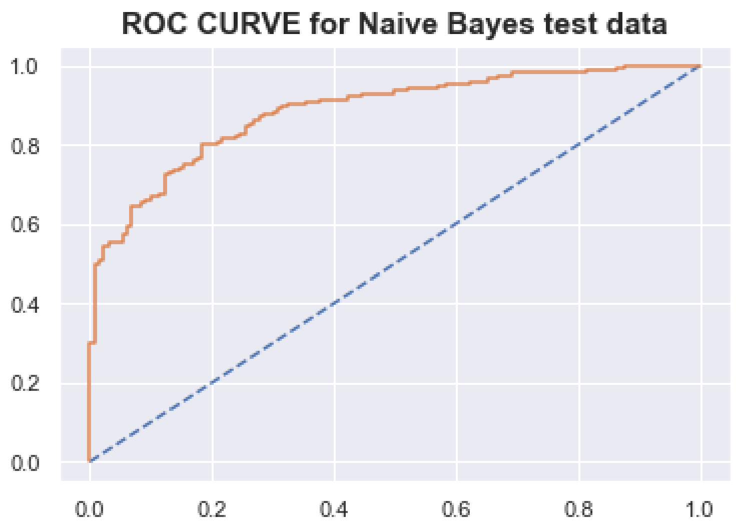


Fig 24: The ROC curve for the testing data using the Naive Bayes model

4. ROC_AUC score
   - Training Data

```
ROC_AUC score: 0.886
```

   - Testing Data

```
ROC_AUC score: 0.885
```

From the above ROC_AUC score and ROC curve, we see that this Naïve Bayes model is a good model having a good ROC_AUC score and steady ROC curve (neglecting the minute noises in the curve) in both train and test data.

5. Classification Report

➢ Training Data

```
              precision    recall  f1-score   support

           0       0.74      0.72      0.73       332
           1       0.88      0.88      0.88       735

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

From the above classification report, we can see that using the Naïve Bayes model, the precision for training data is 0.88, recall is 0.88, f1 score is 0.88 and accuracy is 0.88. A high F1 score is due to high precision and recall scores. A high F1 score also tells that this model was well trained.

➢ Testing Data

```
              precision    recall  f1-score   support

           0       0.68      0.72      0.70       130
           1       0.89      0.87      0.88       328

    accuracy                           0.83       458
   macro avg       0.78      0.79      0.79       458
weighted avg       0.83      0.83      0.83       458
```

From the above classification report, we can see that using the naïve bayes model, the precision for testing data is 0.89, recall is 0.87, f1 score is 0.88 and accuracy is 0.83. Though the accuracy and f1 score is good enough, high recall score is an issue here. But still this model was be considered a valid model.

D. KNN

1. Accuracy

```
Accuracy of the KNN train data for k=13 0.8294283036551078
Accuracy of the KNN test data for k=13 0.8122270742358079
```

Comparing both the train and test accuracies we can say that this model is a valid model

2. Confusion Matrix

➢ Training Data

```
Confusion matrix for the train data for k=13
 [[215 117]
 [ 65 670]]
```

Given above is the confusion matrix for training data using knn model. Here,

- True positive=215, which is actually True (value or positive=1) and has been predicted true too.
- False negative= 117, which is actually True (value or positive=1), but predicted False (value=negative or 0).
- False positive=65, which is actually False (value= negative or 0), but predicted True (1).
- True negative=670, which is actually False and has been predicted False too.

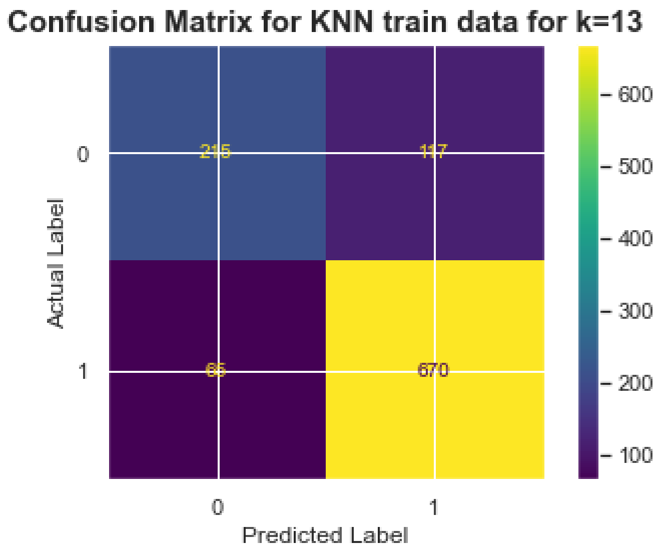


Fig 25: Confusion matrix for the KNN train data where k=13

➢ Testing Data

```
Confusion matrix for the test data for k=13
 [[ 83  47]
 [ 39 289]]
```

Given above is the confusion matrix for testing data using knn model. Here,

- True positive=83, which is actually True (value or positive=1) and has been predicted true too.
- False negative= 47, which is actually True (value or positive=1), but predicted False (value=negative or 0).
- False positive=39, which is actually False (value= negative or 0), but predicted True (1).
- True negative=289, which is actually False and has been predicted False too.

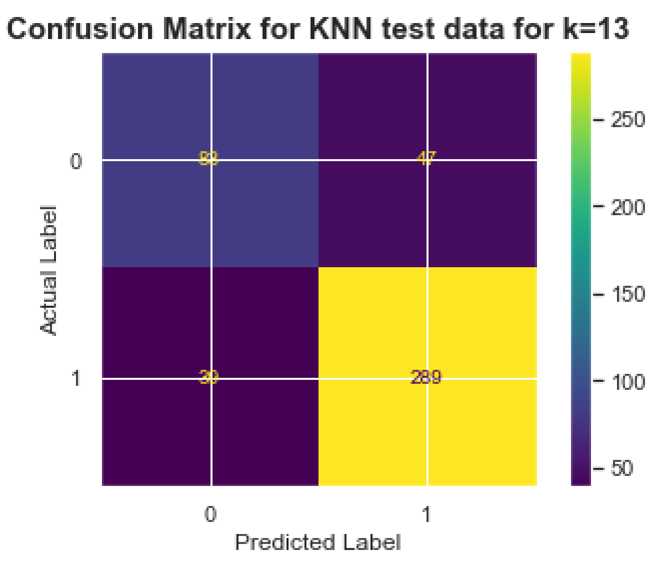


Fig 26: Confusion matrix for the KNN test data where k=13

3. ROC curve

➢ Training Data

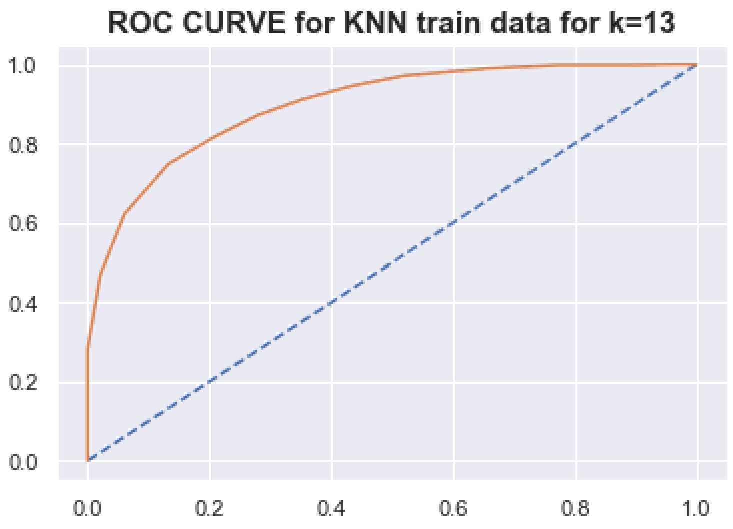


Fig 27: The ROC curve for the training data using the KNN model for k=13

➢ Testing Data

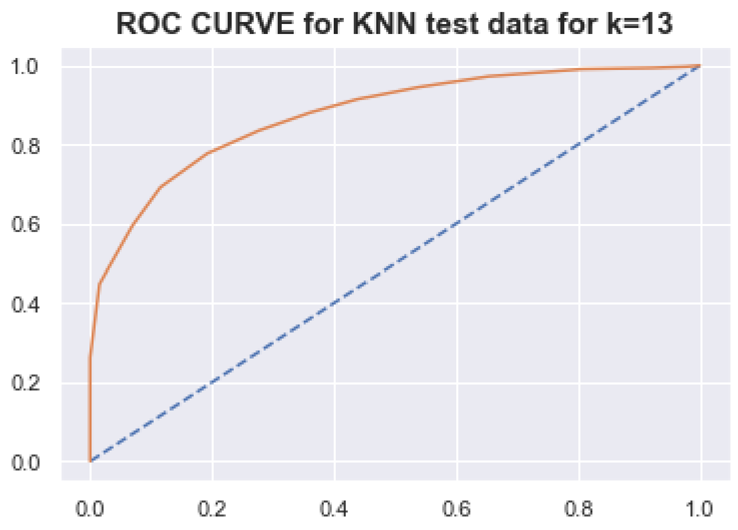


Fig 28: The ROC curve for the testing data using the KNN model for k=13

4. ROC_AUC score
   - Training Data

```
ROC_AUC score: 0.898
```

   - Testing Data

```
ROC_AUC score:0.875
```

From the above ROC_AUC score and ROC curve, we see that this knn model(k=13) is a good model having a good ROC_AUC score and steady ROC curve in both train and test data.

5. Classification Report
   - Training Data

```
              precision    recall  f1-score   support

           0       0.77      0.65      0.70       332
           1       0.85      0.91      0.88       735

    accuracy                           0.83      1067
   macro avg       0.81      0.78      0.79      1067
weighted avg       0.83      0.83      0.83      1067
```

From the above classification report, we can see that using the KNN(k=13) model, the precision for training data is 0.85, recall is 0.91, f1 score is 0.88 and accuracy is 0.83. A high F1 score is due to high precision and recall scores. A high F1 score also tells that this model was well trained.

   - Testing Data

```
              precision    recall  f1-score   support

           0       0.68      0.64      0.66       130
           1       0.86      0.88      0.87       328

    accuracy                           0.81       458
   macro avg       0.77      0.76      0.76       458
weighted avg       0.81      0.81      0.81       458
```

From the above classification report, we can see that using the KNN(k=13) model, the precision for testing data is 0.86, recall is 0.88, f1 score is 0.87 and accuracy is 0.81. Though the accuracy and f1 score is good enough, high recall score is an issue here. But still this model was be considered a valid model.

E. Ada Boosting

1. Accuracy

```
accuracy of train dataset: 0.8472352389878163

accuracy of test dataset: 0.8187772925764192
```

Comparing both the train and test accuracies we can say that this model is not a good model

2. Confusion Matrix

➢ Training Data

```
Confusion matrix for the train data
 [[238  94]
 [ 69 666]]
```

Given above is the confusion matrix for training data using ada boosting model. Here,

- True positive=238, which is actually True (value or positive=1) and has been predicted true too.
- False negative= 94, which is actually True (value or positive=1), but predicted False (value=negative or 0).
- False positive=69, which is actually False (value= negative or 0), but predicted True (1).
- True negative=666, which is actually False and has been predicted False too.

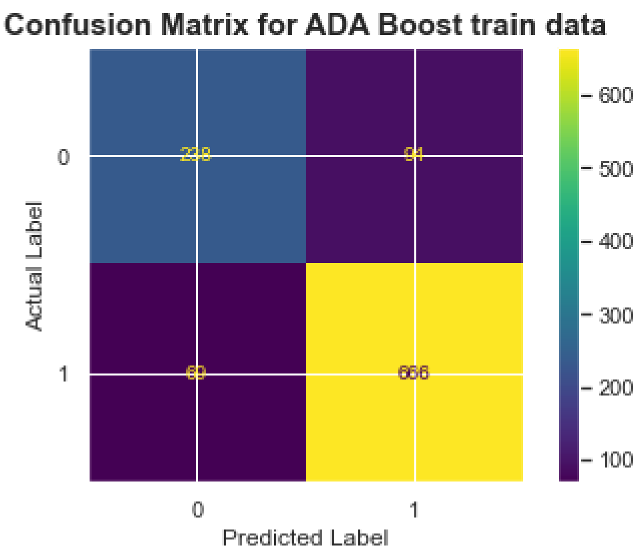


Fig 29: Confusion matrix for the Ada Boosting train data

➢ Testing Data

```
Confusion matrix for the test data
 [[ 90  40]
 [ 43 285]]
```

Given above is the confusion matrix for testing data using ada boosting model. Here,

- True positive=90, which is actually True (value or positive=1) and has been predicted true too.
- False negative= 40, which is actually True (value or positive=1), but predicted False (value=negative or 0).
- False positive=43, which is actually False (value= negative or 0), but predicted True (1).
- True negative=285, which is actually False and has been predicted False too.

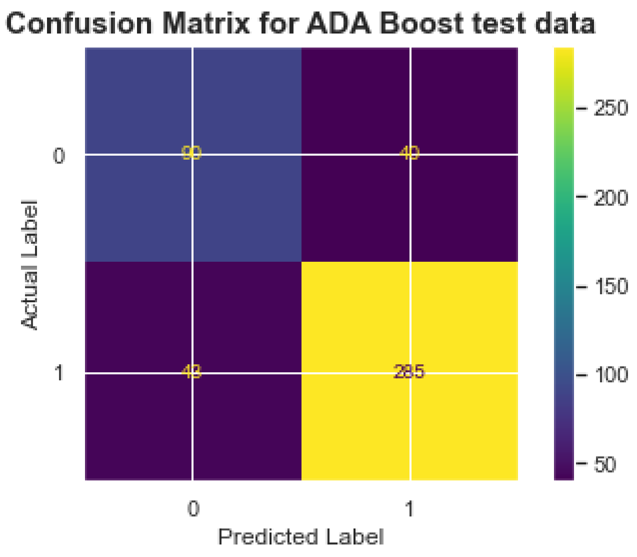


Fig 30: Confusion matrix for Ada Boosting test data

3. ROC curve
   - Training Data

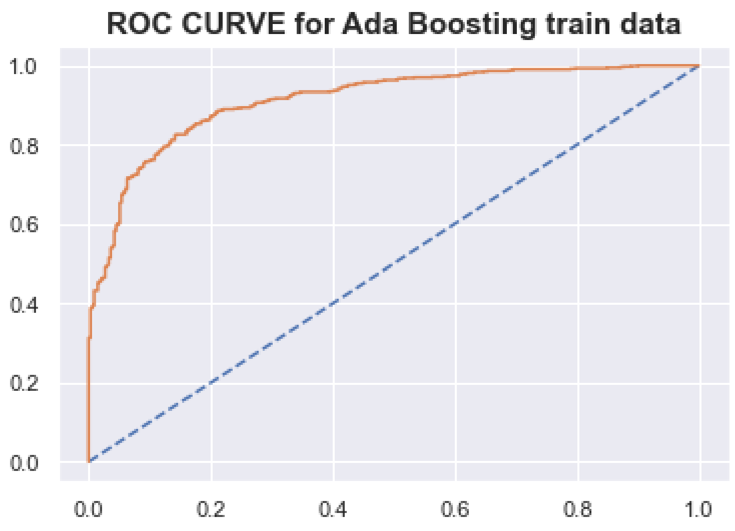


Fig 31: The ROC curve for the training data using the Ada Boosting model

   - Testing Data

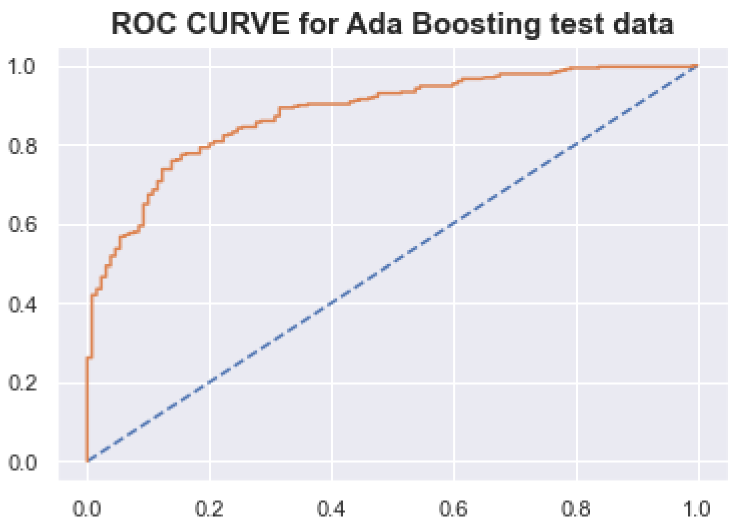


Fig 32: The ROC curve for the testing data using the Ada Boosting model

4. ROC_AUC score
   - Training Data

```
ROC_AUC score: 0.913
```

   - Testing Data

```
ROC_AUC score:0.879
```

From the above ROC_AUC score and ROC curve, we see that this Ada boosting model is not a good model since there is a significant drop in ROC_AUC score and ROC curve when the test data is taken into consideration.

5. Classification Report

➢ Training Data

```
              precision    recall  f1-score   support

           0       0.78      0.72      0.74       332
           1       0.88      0.91      0.89       735

    accuracy                           0.85      1067
   macro avg       0.83      0.81      0.82      1067
weighted avg       0.84      0.85      0.85      1067
```

From the above classification report, we can see that using the Ada Boosting model, the precision for training data is 0.88, recall is 0.91, f1 score is 0.89 and accuracy is 0.85. A high F1 score is due to high precision and recall scores. A high F1 score also tells that this model was well trained.

➢ Testing Data

```
              precision    recall  f1-score   support

           0       0.68      0.69      0.68       130
           1       0.88      0.87      0.87       328

    accuracy                           0.82       458
   macro avg       0.78      0.78      0.78       458
weighted avg       0.82      0.82      0.82       458
```

From the above classification report, we can see that using the Ada Boosting model, the precision for testing data is 0.88, recall is 0.87, f1 score is 0.87 and accuracy is 0.82. Considering the difference between the test-train accuracies and low f1 score in comparison to above created models, this model is not good one.

F. Gradient Boosting

1. Accuracy

```
accuracy of train dataset: 0.8865979381443299

accuracy of test dataset: 0.8318777292576419
```

Comparing both the train and test accuracies we can say that this model is not a good model

2. Confusion Matrix

➢ Training Data

```
Confusion matrix for the train data
 [[262  70]
 [ 51 684]]
```

Given above is the confusion matrix for training data using gradient boosting model. Here,

- True positive=262, which is actually True (value or positive=1) and has been predicted true too.
- False negative= 70, which is actually True (value or positive=1), but predicted False (value=negative or 0).
- False positive=51, which is actually False (value= negative or 0), but predicted True (1).
- True negative=684, which is actually False and has been predicted False too.

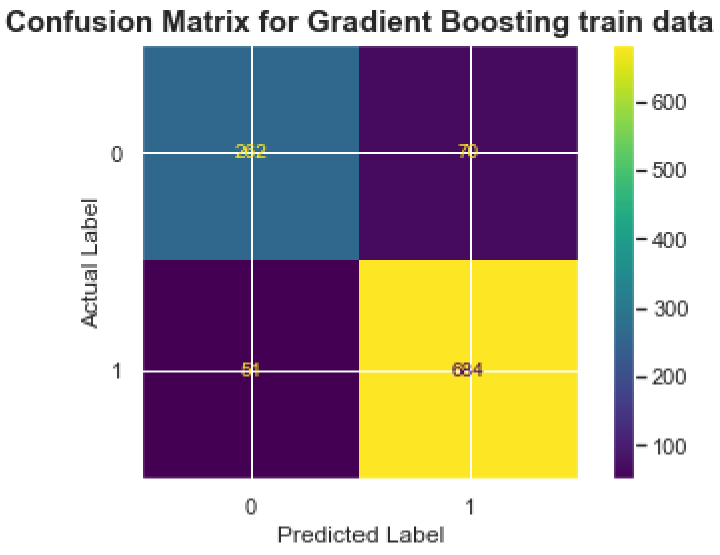


Fig 33: Confusion matrix for the Gradient boosting train data

➢ Testing Data

```
Confusion matrix for the test data
 [[ 96  34]
 [ 43 285]]
```

Given above is the confusion matrix for testing data using gradient boosting model. Here,

- True positive=96, which is actually True (value or positive=1) and has been predicted true too.
- False negative= 34, which is actually True (value or positive=1), but predicted False (value=negative or 0).
- False positive=43, which is actually False (value= negative or 0), but predicted True (1).
- True negative=285, which is actually False and has been predicted False too.

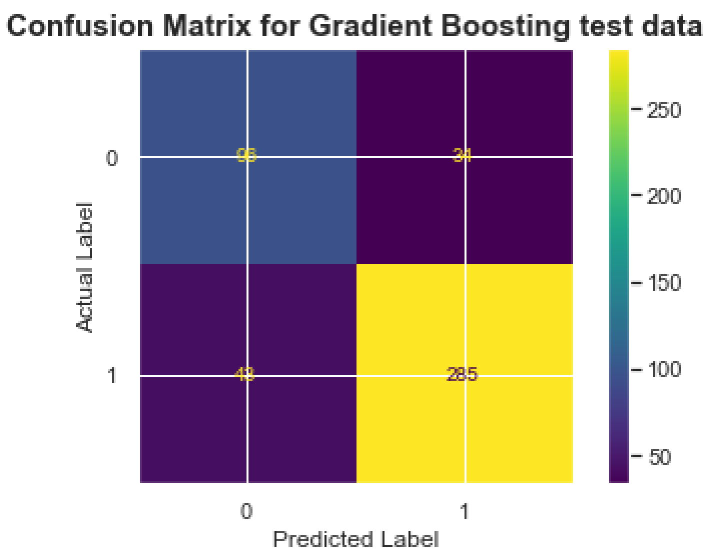


Fig 34: Confusion matrix for the Gradient Boosting test data

3. ROC curve
   - Training Data

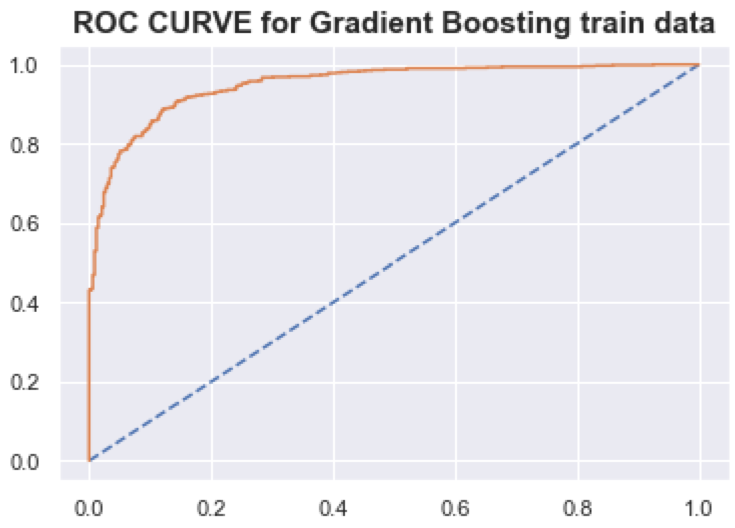


Fig 35: The ROC curve for the training data using the gradient boosting model

   - Testing Data

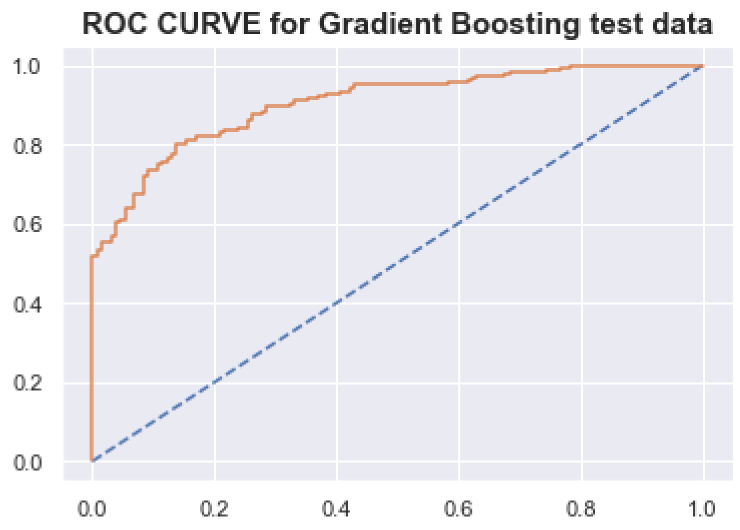


Fig 36: The ROC curve for the testing data using the gradient boosting model

4. ROC_AUC score
   - Training Data

```
ROC_AUC score: 0.950
```

   - Testing Data

```
ROC_AUC score :0.904
```

From the above ROC_AUC score and ROC curve, we see that this gradient boosting model is not a good model since there is a significant drop in ROC_AUC score and ROC curve when the test data is taken into consideration.

5. Classification Report
   - Training Data

```
              precision    recall  f1-score   support

           0       0.84      0.79      0.81       332
           1       0.91      0.93      0.92       735

    accuracy                           0.89      1067
   macro avg       0.87      0.86      0.87      1067
weighted avg       0.89      0.89      0.89      1067
```

From the above classification report, we can see that using the Gradient Boosting model, the precision for training data is 0.91, recall is 0.93, f1 score is 0.92 and accuracy is 0.89. A high F1 score is due to high precision and recall scores. A high F1 score also tells that this model was well trained.

   - Testing Data

```
              precision    recall  f1-score   support

           0       0.69      0.74      0.71       130
           1       0.89      0.87      0.88       328

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

From the above classification report, we can see that using the Gradient Boosting model, the precision for testing data is 0.89, recall is 0.87, f1 score is 0.88 and accuracy is 0.83. Considering the difference between the test-train accuracies and low f1 score in comparison to above created models, this model is not good one.

G. Bagging (along with Random Forest)

1. Accuracy

```
Accuracy on Bagging train data: 0.9653233364573571
```

```
Accuracy on Bagging test data: 0.8318777292576419
```

Comparing both the train and test accuracies we can say that this model is a not good model.

2. Confusion Matrix

➢ Training Data

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

Given above is the confusion matrix for training data using Bagging model. Here,

- True positive=304, which is actually True (value or positive=1) and has been predicted true too.
- False negative= 28, which is actually True (value or positive=1), but predicted False (value=negative or 0).
- False positive=9, which is actually False (value= negative or 0), but predicted True (1).
- True negative=726, which is actually False and has been predicted False too.

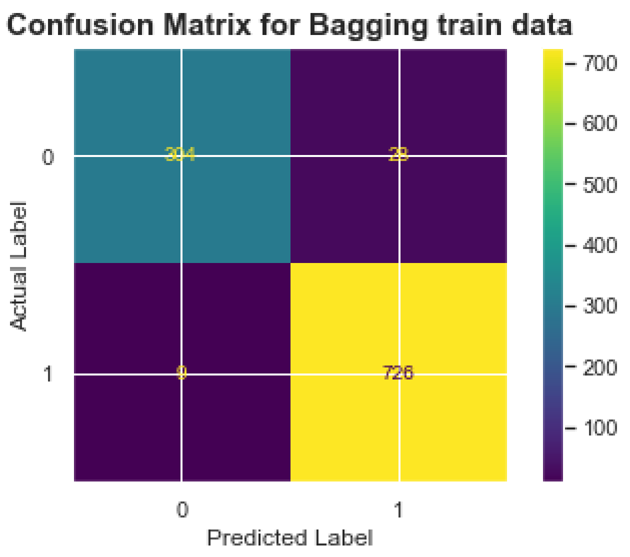


Fig 37: Confusion matrix for the Bagging train data

➢ Testing Data

```
[[ 91  39]
 [ 38 290]]
```

Given above is the confusion matrix for testing data using Bagging model. Here,

- True positive=91, which is actually True (value or positive=1) and has been predicted true too.
- False negative= 39, which is actually True (value or positive=1), but predicted False (value=negative or 0).
- False positive=38, which is actually False (value= negative or 0), but predicted True (1).
- True negative=290, which is actually False and has been predicted False too.

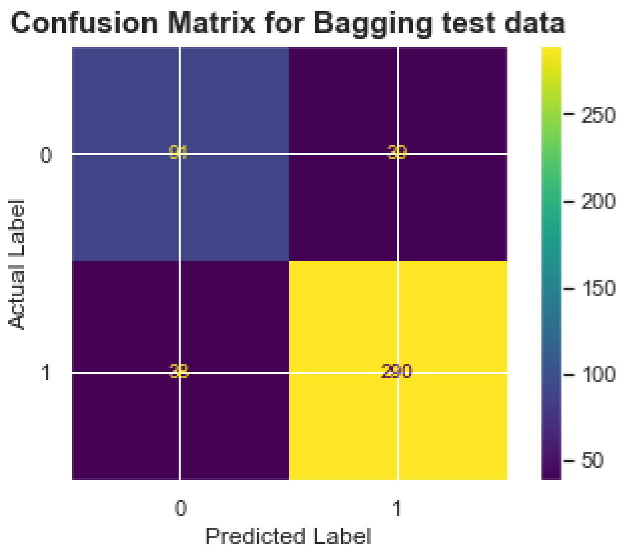


Fig 38: Confusion matrix for the Bagging test data

3. ROC curve
   - Training Data

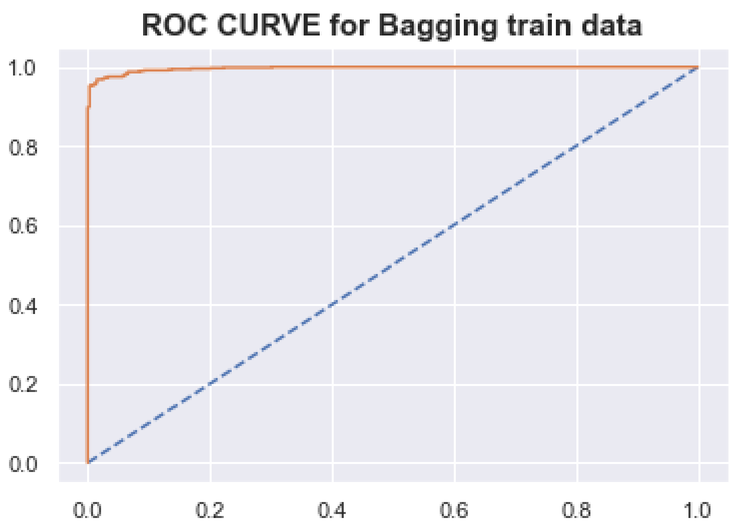


Fig 39: The ROC curve for the training data using the bagging model

   - Testing Data

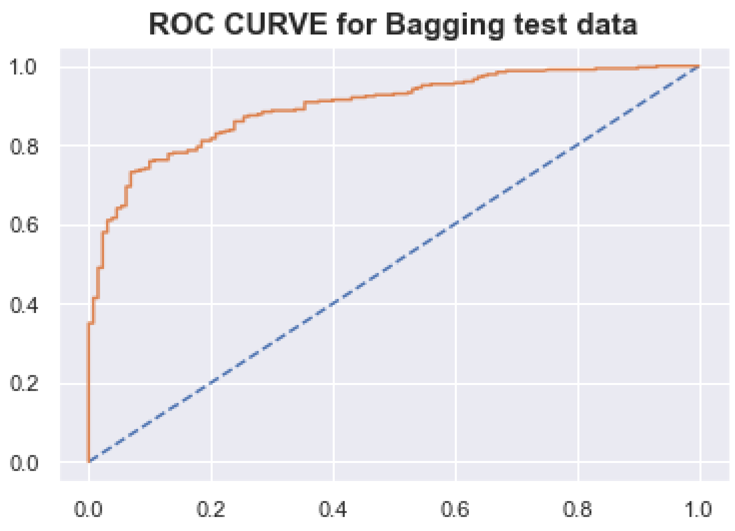


Fig 40: The ROC curve for the testing data using the bagging model

4. ROC_AUC score
   - Training Data

```
ROC_AUC score: 0.997
```

➢ Testing Data

```
ROC_AUC score :0.897
```

From the above ROC_AUC score and ROC curve, we see that this bagging model is not a good model since there is a significant drop in ROC_AUC score and ROC curve when the test data is taken into consideration.

5. Classification Report

➢ Training Data

```
              precision    recall  f1-score   support

           0       0.97      0.92      0.94       332
           1       0.96      0.99      0.98       735

    accuracy                           0.97      1067
   macro avg       0.97      0.95      0.96      1067
weighted avg       0.97      0.97      0.97      1067
```

From the above classification report, we can see that using the Bagging model, the precision for training data is 0.96, recall is 0.99, f1 score is 0.98 and accuracy is 0.97. A high F1 score is due to high precision and recall scores. A high F1 score also tells that this model was well trained.

➢ Testing Data

```
              precision    recall  f1-score   support

           0       0.71      0.70      0.70       130
           1       0.88      0.88      0.88       328

    accuracy                           0.83       458
   macro avg       0.79      0.79      0.79       458
weighted avg       0.83      0.83      0.83       458
```

From the above classification report, we can see that using the Bagging model, the precision for testing data is 0.88, recall is 0.88, f1 score is 0.88 and accuracy is 0.83. Considering the difference between the test-train accuracies and low f1 score in comparison to above created models, this model is not good one.

MODEL COMPARISON

MODELS	DATA TYPE	ACCURACY	F1 SCORE	PRECISON	RECALL	AUC SCORE
LOGISTIC REGRESSION	TRAIN	84	89	87	91	0.889
	TEST	82	88	87	89	0.882
LINEAR DISCRIMINANT ANALYSIS	TRAIN	84	88	87	90	0.889
	TEST	82	87	87	88	0.884
NAÏVE BAYES	TRAIN	83	88	88	88	0.886
	TEST	83	88	89	87	0.885
KNN	TRAIN	83	88	85	91	0.898
	TEST	81	87	86	88	0.875
ADA BOOSTING	TRAIN	85	89	88	91	0.913
	TEST	82	87	88	87	0.879
GRADIENT BOOSTING	TRAIN	89	92	91	93	0.950
	TEST	83	88	89	87	0.904
BAGGING ON RANDOM FOREST	TRAIN	97	98	96	99	0.997
	TEST	83	88	88	88	0.897

Table 13: MODEL COMPARISON

Comparing all 7 created models, I found Naïve Bayes model to be the best. Naïve Bayes has a steady accuracy and AUC for both train and test data along with high precision and f1 score. Moreover, when 10-fold cross validation was done on first 4 models in the table above(since they have performed approximately similar on both train and test data), the scores both on train and test data set respectively for all 10 folds are almost same for Naïve Bayes, proving it to be the most valid model. Thus, it was proved that Naïve Bayes model is the best among all.

Q1.8: Based on your analysis and working on the business problem, detail out appropriate insights and recommendations to help the management solve the business objective. There should be at least 3-4 Recommendations and insights in total. Recommendations should be easily understandable and business specific, students should not give any technical suggestions. Full marks should only be allotted if the recommendations are correct and business specific.

My insights from carrying out Exploratory data analysis and building models to predict the outcome on the given data are as follows:

• The Labour party leader Blair stands more chances of winning the elections when compared to Hague. Hague should invent new ways to promote himself as a potential leader and try more to convince the people about it.

• The voters of conservative party have more Eurosceptic sentiment, so if the party promotes this sentiment during their campaign, they are more likely to achieve more voters who support this ideology.

• Both the current national economic conditions and household economic

conditions are at an average level right now, So there is a need for a lot of

reforms to be developed to improve these conditions. Both the political parties should bring in new strategies to improve the conditions and propagate it to their voters efficiently.

• The majority of age groups that vote seem to come from an average of 50, this

means that votes from the younger demographic seem to low. So the party that

is successful in winning the hearts of this younger demographic of voters has a

good chance of turning the tide of the election results.

• The voters of Labour party has better knowledge when compared to Conservative party when it comes to economical conditions of both political and household realms. So, the conservative party should give more realistic insight to their voters on the present situations and wisely promote their propaganda.

• Voters of Labour party has more knowledge about European Integration than the voters of conservative party. So, the conservative party is advised to take political classes for common people on the same subject or issue booklets or articles educating its voters on the same issue.

- Conservative party has to work more to win this election since Hague doesn't enjoy as much popularity as Blair or they should find a better candidate than Hague in order to win this election.

TEXT ANALYTICS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

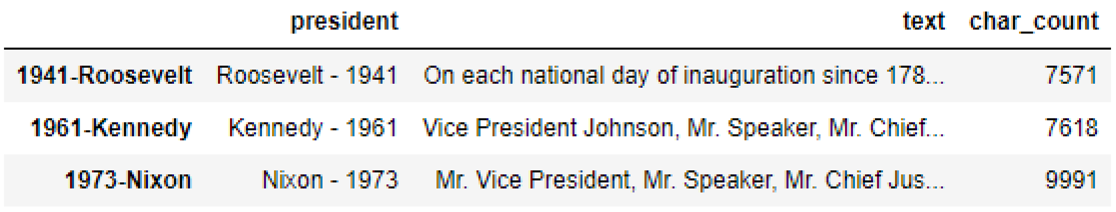
In this particular project, we are going to work on the inaugural corpora from the nltk in Python. We will be looking at the following speeches of the Presidents of the United States of America:

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

INTRODUCTION

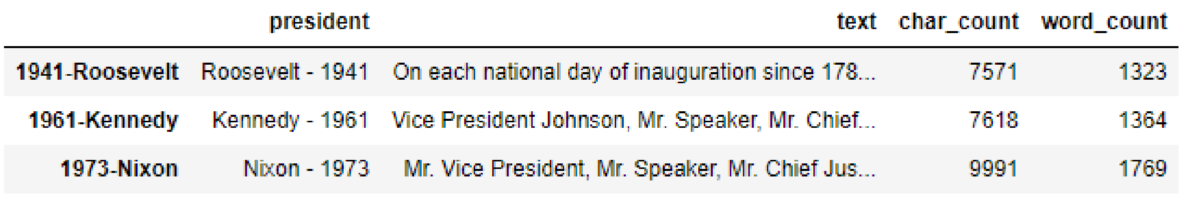
The purpose of this whole exercise is to various techniques involved in text analytics to read the test data and do various operations on the text data such as counting the number of words, sentences, characters etc. We can even do sentimental analysis on the data after learning the nature of the text.

Q2.1. Find the number of characters, words and sentences for the mentioned documents.



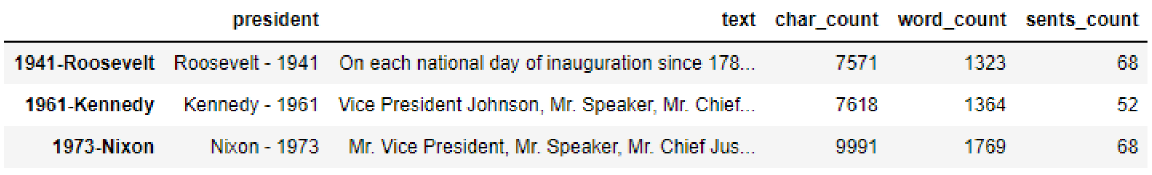
	president	text	char_count
1941-Roosevelt	Roosevelt - 1941	On each national day of inauguration since 178...	7571
1961-Kennedy	Kennedy - 1961	Vice President Johnson, Mr. Speaker, Mr. Chief...	7618
1973-Nixon	Nixon - 1973	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	9991

Table 1. Table showing the character count in the given 3 texts.



	president	text	char_count	word_count
1941-Roosevelt	Roosevelt - 1941	On each national day of inauguration since 178...	7571	1323
1961-Kennedy	Kennedy - 1961	Vice President Johnson, Mr. Speaker, Mr. Chief...	7618	1364
1973-Nixon	Nixon - 1973	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	9991	1769

Table 2. Table showing the word count in the given 3 texts.

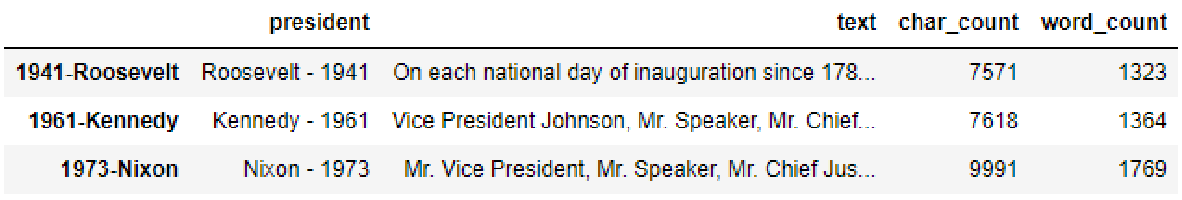


	president	text	char_count	word_count	sents_count
1941-Roosevelt	Roosevelt - 1941	On each national day of inauguration since 178...	7571	1323	68
1961-Kennedy	Kennedy - 1961	Vice President Johnson, Mr. Speaker, Mr. Chief...	7618	1364	52
1973-Nixon	Nixon - 1973	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	9991	1769	68

Table 3. Table showing the sentence count in the given 3 texts.

Q2.2. 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.

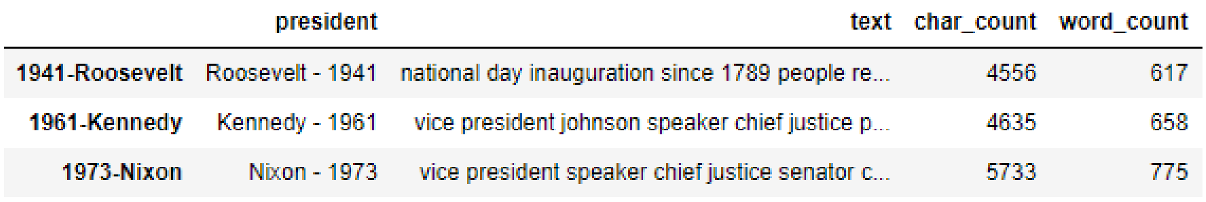
Before the removal of stopwords



	president	text	char_count	word_count
1941-Roosevelt	Roosevelt - 1941	On each national day of inauguration since 178...	7571	1323
1961-Kennedy	Kennedy - 1961	Vice President Johnson, Mr. Speaker, Mr. Chief...	7618	1364
1973-Nixon	Nixon - 1973	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	9991	1769

Table 4. Table showing the word count in the given 3 texts before removal of stop words.

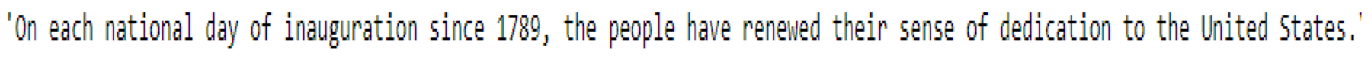
After the removal of stopwords



	president	text	char_count	word_count
1941-Roosevelt	Roosevelt - 1941	national day inauguration since 1789 people re...	4556	617
1961-Kennedy	Kennedy - 1961	vice president johnson speaker chief justice p...	4635	658
1973-Nixon	Nixon - 1973	vice president speaker chief justice senator c...	5733	775

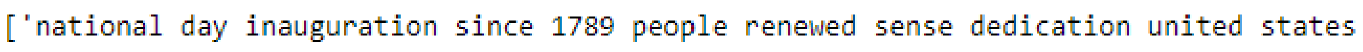
Table 5. Table showing the word count in the given 3 texts after removal of stop words.

First sentence before the removal of stop words from Roosevelt speech



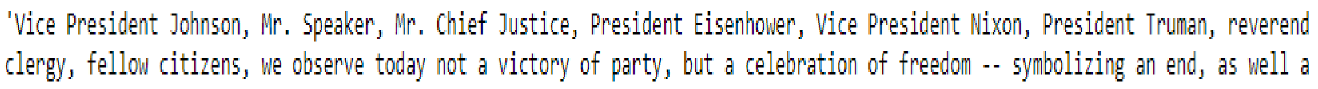
```
'On each national day of inauguration since 1789, the people have renewed their sense of dedication to the United States.'
```

First sentence after the removal of stop words from Roosevelt speech



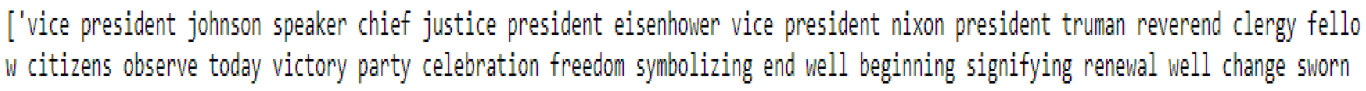
```
['national day inauguration since 1789 people renewed sense dedication united states
```

First sentence before the removal of stop words from Kennedy speech



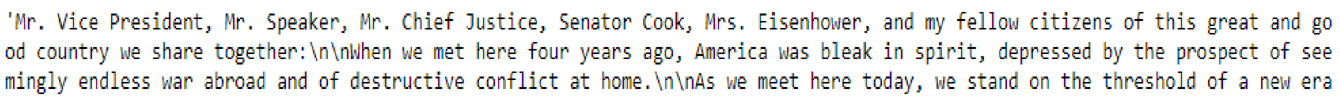
```
'Vice President Johnson, Mr. Speaker, Mr. Chief Justice, President Eisenhower, Vice President Nixon, President Truman, reverend
clergy, fellow citizens, we observe today not a victory of party, but a celebration of freedom -- symbolizing an end, as well a
```

First sentence after the removal of stop words from Kennedy speech



```
['vice president johnson speaker chief justice president eisenhower vice president nixon president truman reverend clergy fello
w citizens observe today victory party celebration freedom symbolizing end well beginning signifying renewal well change sworn
```

First sentence before the removal of stop words from Nixon speech



```
'Mr. Vice President, Mr. Speaker, Mr. Chief Justice, Senator Cook, Mrs. Eisenhower, and my fellow citizens of this great and go
od country we share together:\n\nWhen we met here four years ago, America was bleak in spirit, depressed by the prospect of see
mingly endless war abroad and of destructive conflict at home.\n\nAs we meet here today, we stand on the threshold of a new era
```

First sentence after the removal of stop words from Nixon speech

```
['vice president speaker chief justice senator cook mrs eisenhower fellow citizens great good country share together met four y
ears ago america bleak spirit depressed prospect seemingly endless war abroad destructive conflict home meet today stand thresh
```

Q2.3. Which word occurs the most number of times in his inaugural address for each president? Mention the top three words. (after removing the stopwords)

```
Top three words in the 1941-Roosevelt speech

nation       11
know         10
democracy     9
dtype: int64
```

```
Top three words in the 1961-Kennedy speech

sides    8
world    8
new      7
dtype: int64
```

```
Top three words in the  1973-Nixon speech

peace    19
world    16
new      15
dtype: int64
```

Q2.4. Plot the word cloud of each of the three speeches. (after removing the stopwords)

1941-Roosevelt speech word cloud

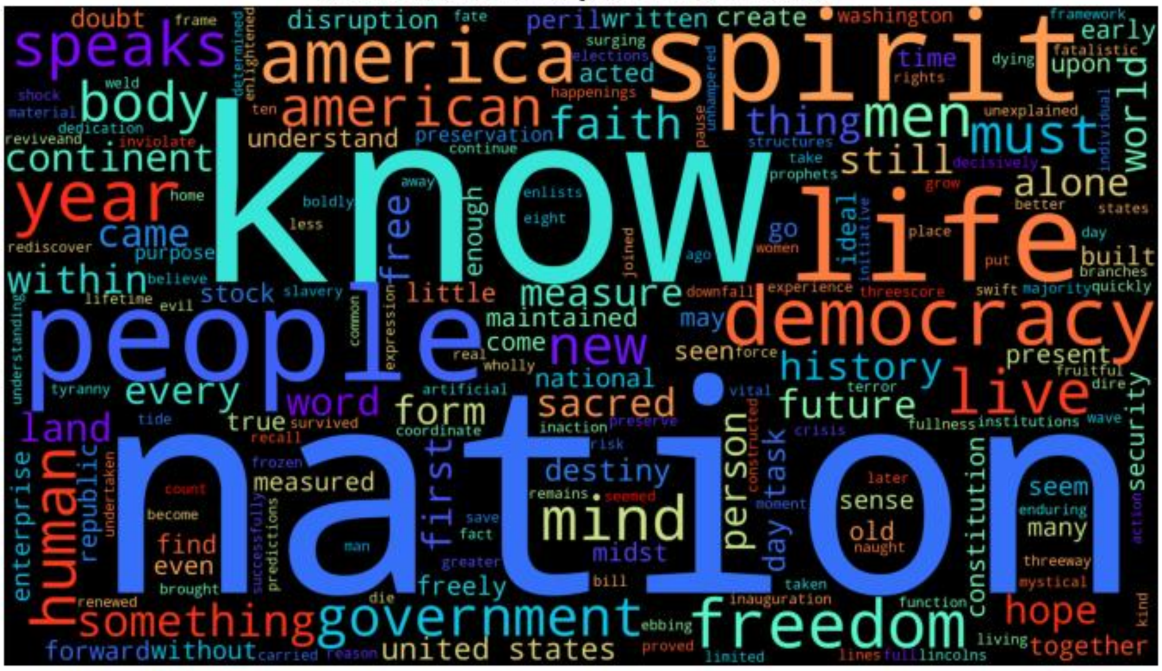


Fig 1: The figure shows the word cloud of 1941-Roosevelt speech

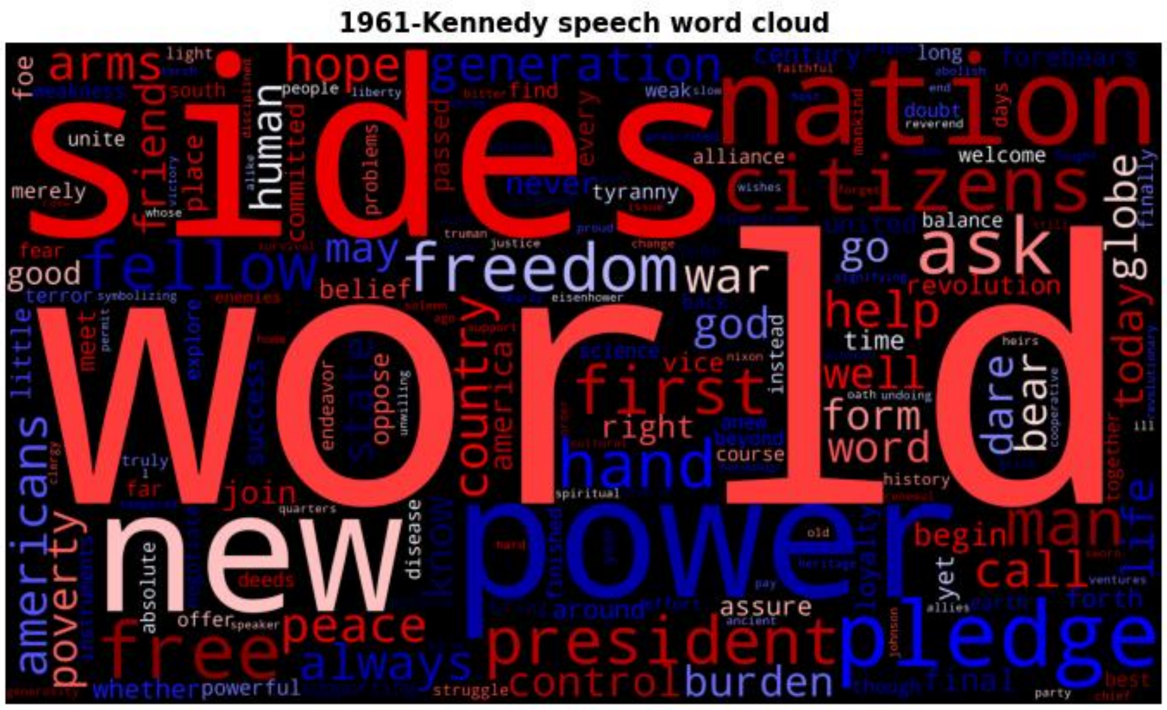


Fig 2: The figure shows the word cloud of 1961-Kennedy speech

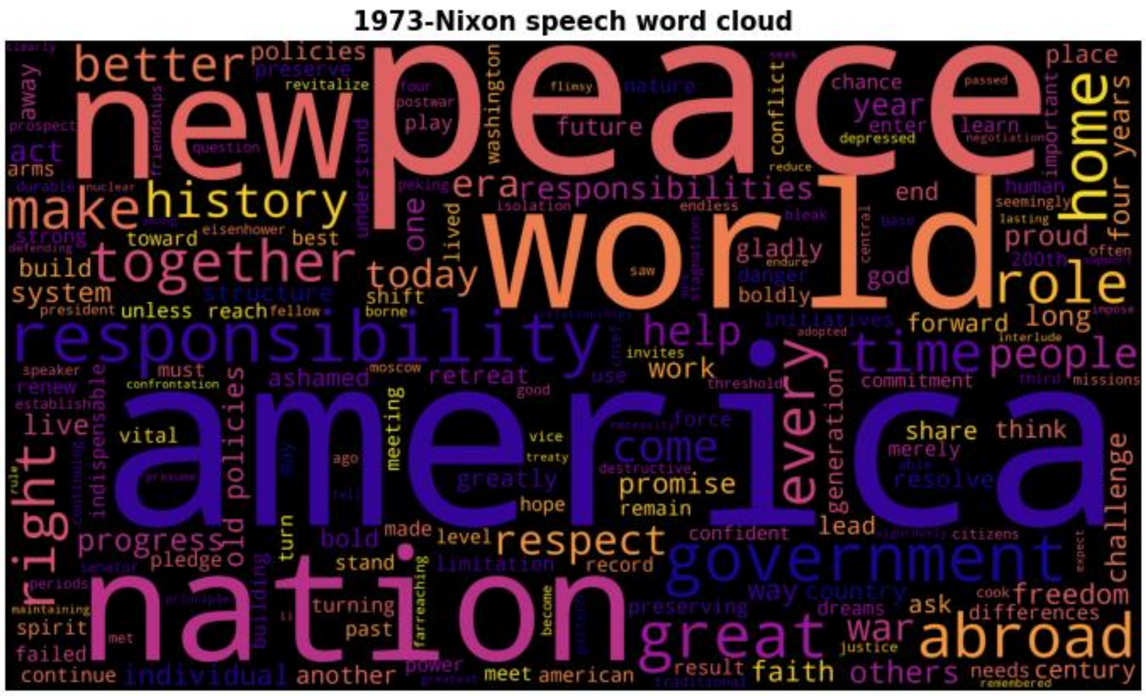


Fig 3: The figure shows the word cloud of 1973-Nixon speech

THE END