# MACHINE LEARNING

## PROJECT REPORT

YOGESH NEGI

## Problem 1:

You are hired by one of the leading news channels CNBE who wants to analyze recent elections. This survey was conducted on 1525 voters with 9 variables. You have to build a model, to predict which party a voter will vote for on the basis of the given information, to create an exit poll that will help in predicting overall win and seats covered by a particular party.

1.1 Read the dataset. 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.

## Data Information:

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

## Observation:

- The dataset had 8 duplicated values. So, we are dropped them.
- The data set had 1525 rows and 9 columns. After dropping the duplicate values, there are 1517 rows and 9columns.
- It has 7 numerical data types and 2 categorical data types.
- There is no null value in any column.

## Checking for missing values:

```
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
```

There are no missing values.

## Checking for duplicated values:

```
8
```

There are 8 duplicated values. So, we are dropping them.

## Data description:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| age | 1517.0 | 54.241266 | 15.701741 | 24.0 | 41.0 | 53.0 | 67.0 | 93.0 |
| economic.cond.national | 1517.0 | 3.245221 | 0.881792 | 1.0 | 3.0 | 3.0 | 4.0 | 5.0 |
| economic.cond.household | 1517.0 | 3.137772 | 0.931069 | 1.0 | 3.0 | 3.0 | 4.0 | 5.0 |
| Blair | 1517.0 | 3.335531 | 1.174772 | 1.0 | 2.0 | 4.0 | 4.0 | 5.0 |
| Hague | 1517.0 | 2.749506 | 1.232479 | 1.0 | 2.0 | 2.0 | 4.0 | 5.0 |
| Europe | 1517.0 | 6.740277 | 3.299043 | 1.0 | 4.0 | 6.0 | 10.0 | 11.0 |
| political.knowledge | 1517.0 | 1.540541 | 1.084417 | 0.0 | 0.0 | 2.0 | 2.0 | 3.0 |

## Checking the skewness of the data:

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

## The rule of thumb of skewness seems to be:

- If the skewness is between -0.5 and 0.5, the data are fairly symmetrical.
- If the skewness is between -1 and – 0.5 or between 0.5and 1, the data are moderately skewed.
- If the skewness is less than -1 or greater than 1, the data are highly skewed.
- The Age and Time_of_Vote columns are slightly positively skewed, while the Income column is heavily positively skewed. This suggests that there may be some outliers or extreme values in the Income column.

## Insights:

- Here, we can see that there isn't much skewness in the data. All the values seems to be between -0.5 and 0.5.
- The value of 'Blair' is a little bit higher than -0.5.
- The data overall, is fairly symmetrical.

1.2 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. 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.

## Exploratory Data Analysis:

## Null value check:

```
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
```

There are no null values present in the data.

## Data types:

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

There are 7 numerical and 2 categorical data types in the data.

## Shape of the data:

```
no. of rows: 1517
no. of columns: 9
```

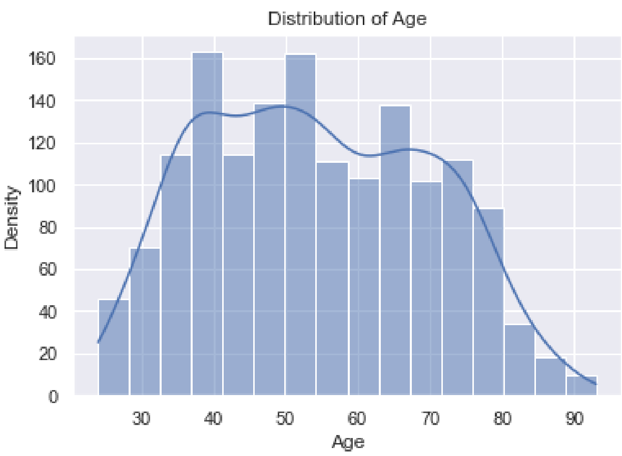
There are 1517 rows and 9 columns in the data.

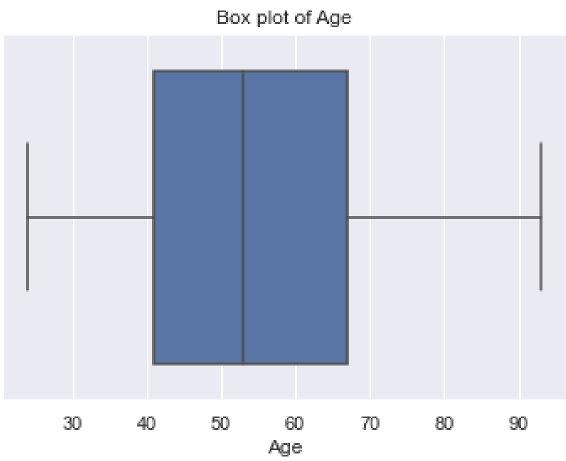
## Univariate Analysis:

## Description of 'age':

```
count    1517.000000
mean       54.241266
std        15.701741
min        24.000000
25%        41.000000
50%        53.000000
75%        67.000000
max        93.000000
Name: age, dtype: float64
```

## Histogram and box plot of 'age':

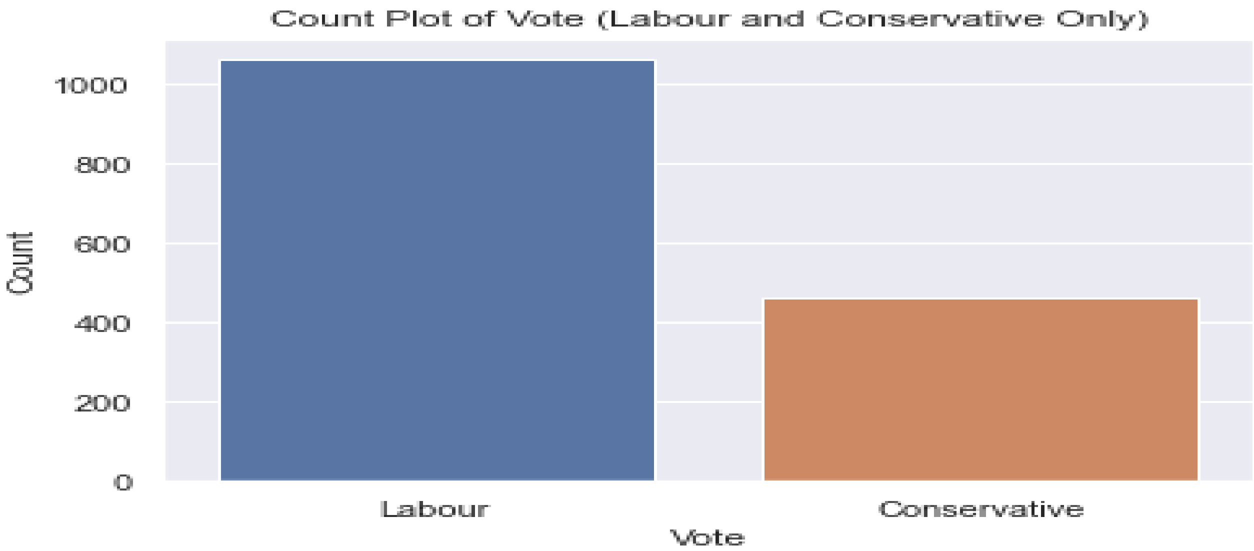


## Observation:

- The data is normally distributed.
- Maximum number of people are aged between 40 and 70.
- Outliers are not present.
- The minimum value is 24 and the maximum value is 93.
- The mean value is 54.241266

## Count plot of 'vote':

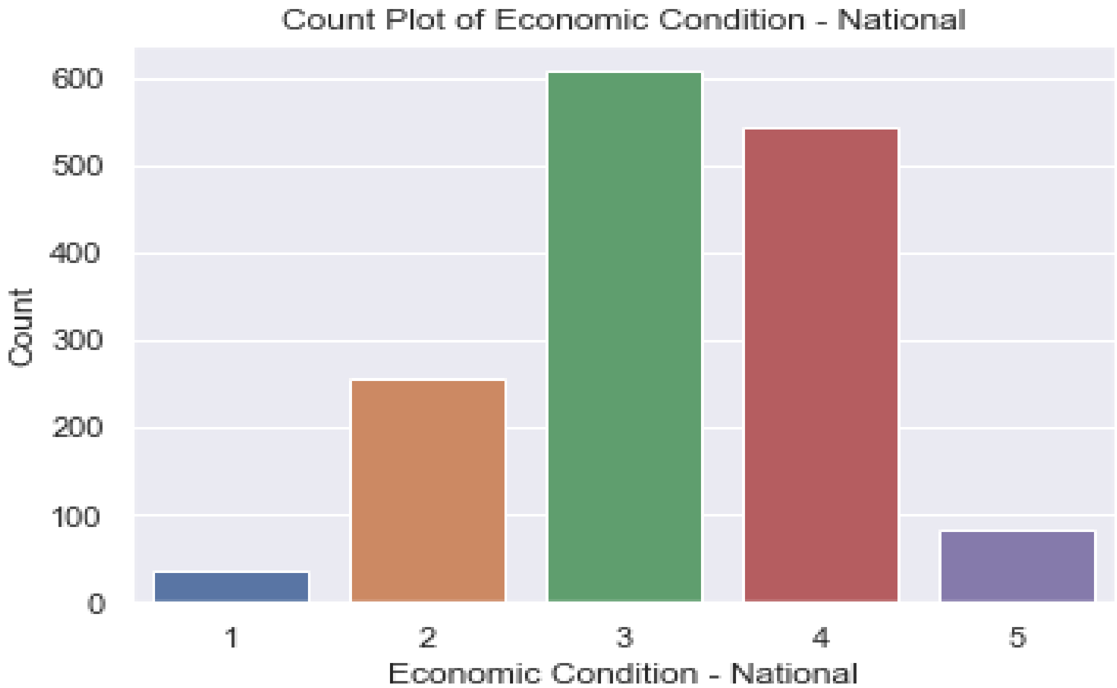


## Viewing the exact values of the variables of 'vote':

Labour 1063
Conservative 462
Name: vote, dtype: int64

## Observation:

- Labour party has higher number of votes. It has more than double the votes of conservative party.
- Labour party has 1063 votes.
- Conservative party has 462 votes.



## Count plot of 'economic.cond.national':

## Viewing the exact values of the variables of 'economic.cond.national':

```
3    607
4    542
2    257
5     82
1     37
Name: economic.cond.national, dtype: int64
```

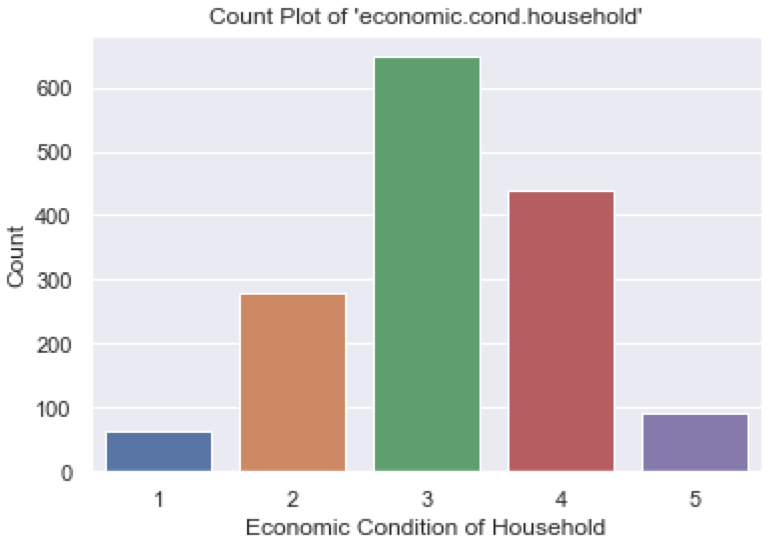
## Mean of 'economic.cond.national':

Mean of 'economic.cond.national': 3.2459016393442623

## Observation:

- The top 2 variables are 3 and 4.
- 1 has the least value which is 37.
- 3 has the highest value which is 607.
- 3 is slightly higher than the 2nd highest variable 4 whose value is 542.
- The average score of 'economic.cond.national' is 3.2459016393442623

## Count plot of 'economic.cond.household':




## Viewing the exact values of the variables of 'economic.cond.household':

3    648
4    440
2    280
5     92
1     65
Name: economic.cond.household, dtype: int64

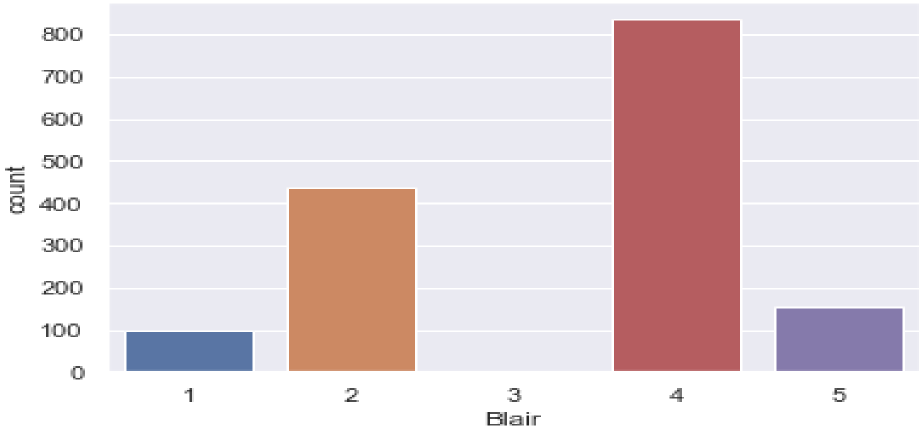
## Mean of 'economic.cond.household':

Mean of "economic.cond.household": 2.140327868852459

## Observation:

- The top 2 variables are 3 and 4.
- 1 has the least value which is 65.
- 3 has the highest value which is 648.
- 3 is moderately higher than the 2nd highest variable 4whose value is 440.
- The average score of 'economic.cond.household' is 2.14032786885245

## Count plot of 'Blair':




## Viewing the exact values of the variables of 'Blair':

4    836
2    438
5    153
1     97
3      1
Name: Blair, dtype: int64

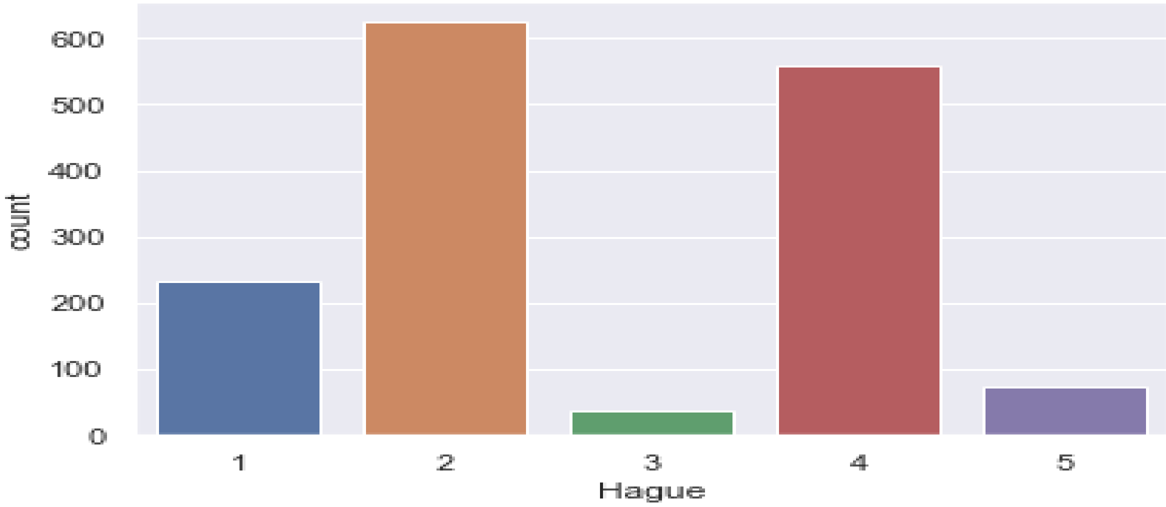
## Mean of 'Blair':

Mean of Blair: 3.3344262295081966

## Observation:

- The top 2 variables are 2 and 4.
- 3 has the least value which is 1.
- 4 has the highest value which is 836.
- 4 is much higher than the 2nd highest variable 2 whose value is 438.
- The average score of 'Blair' is 3.3344262295081966

## Count plot of 'Hague':




## Viewing the exact values of the variables of 'Hague':

```
2    624
4    558
1    233
5     73
3     37
Name: Hague, dtype: int64
```

## Mean of 'Hague':

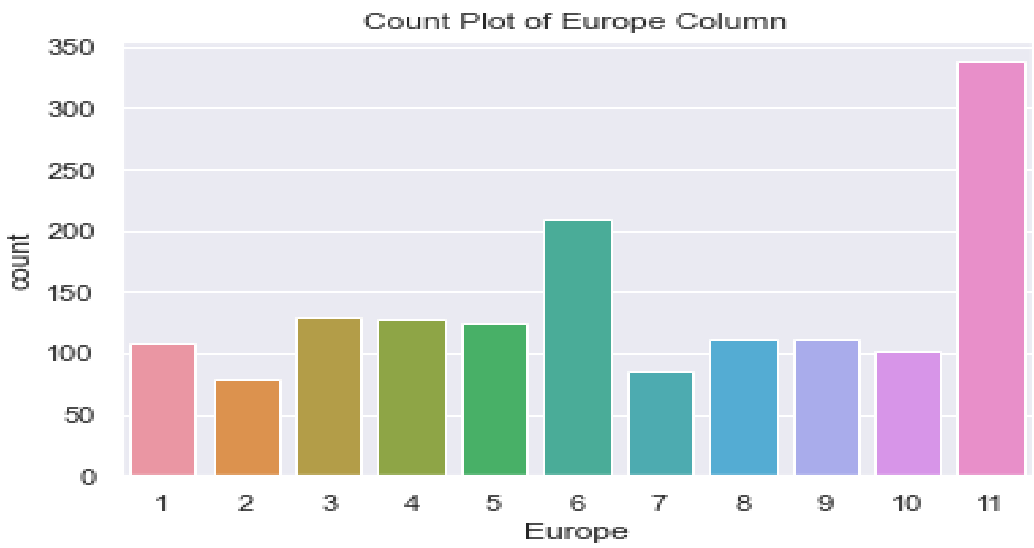
```
Mean of Hague column: 2.7468852459016393
```

## Observation:

- The top 2 variables are 2 and 4.
- 3 has the least value which is 37.
- 2 has the highest value which is 624.

- 2 is slightly higher than the 2nd highest variable 4 whosevalue is 558.
- The average score of 'Blair' is 2.74688

## Count plot of 'Europe':




## Viewing the exact values of the variables of 'Europe':

```
11    338
6     209
3     129
4     127
5     124
8     112
9     111
1     109
10    101
7      86
2      79
Name: Europe, dtype: int64
```

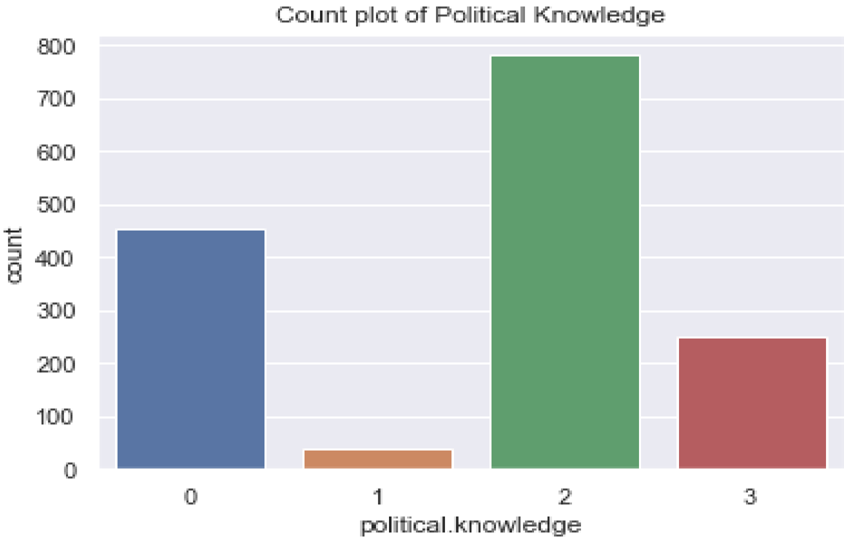
## Mean of 'Europe':

Mean of 'Europe':  6.728524590163935

## Observation:

- The top 2 variables are 11 and 6.
- 2 has the least value which is 79.
- 11 has the highest value which is 338.
- 11 is moderately higher than the 2nd highest variable 6whose value is 209.
- The average score of 'Europe' is 6.728524

## Count plot of 'political.knowledge':




## Viewing the exact values of the variables of 'political.knowledge':

```
2    782
0    455
3    250
1     38
Name: political.knowledge, dtype: int64
```

## Mean of 'Europe':

```
Mean of political.knowledge: 1.5422950819672132
```

## Observation:

- The top 2 variables are 2 and 0.

- 1 has the least value which is 38.
- 2 has the highest value which is 782.
- 2 is much higher than the 2nd highest variable 0 whose value is 455.
- We can see that, 455 out of 1517 people do not have any knowledge of parties' positions on European integration which is 29.93% of the total population.
- The average score of ' political.knowledge ' is 1.542295081

## Bivariate Analysis:

## Strip plot of 'vote' and 'age':

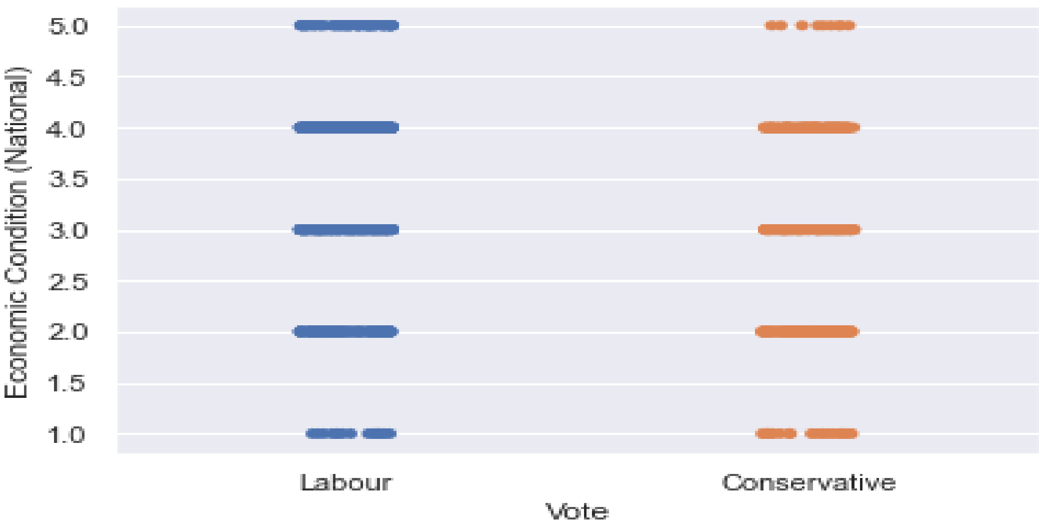

## Viewing the exact values of the variables of 'vote'with respect to 'gender':

```
vote          gender
Conservative  female    259
              male      203
Labour        female    553
              male      510
Name: gender, dtype: int64
```

## Observation:

- We can clearly see that, the labour party has got more votes than the conservative party.
- In every age group, the labour party has got more votes than the conservative party.
- Female votes are considerably higher than the male votes in both parties.
- In both genders, the labour party has got more votes than the conservative party.

## Strip plot of 'vote' and 'economic.cond.national':



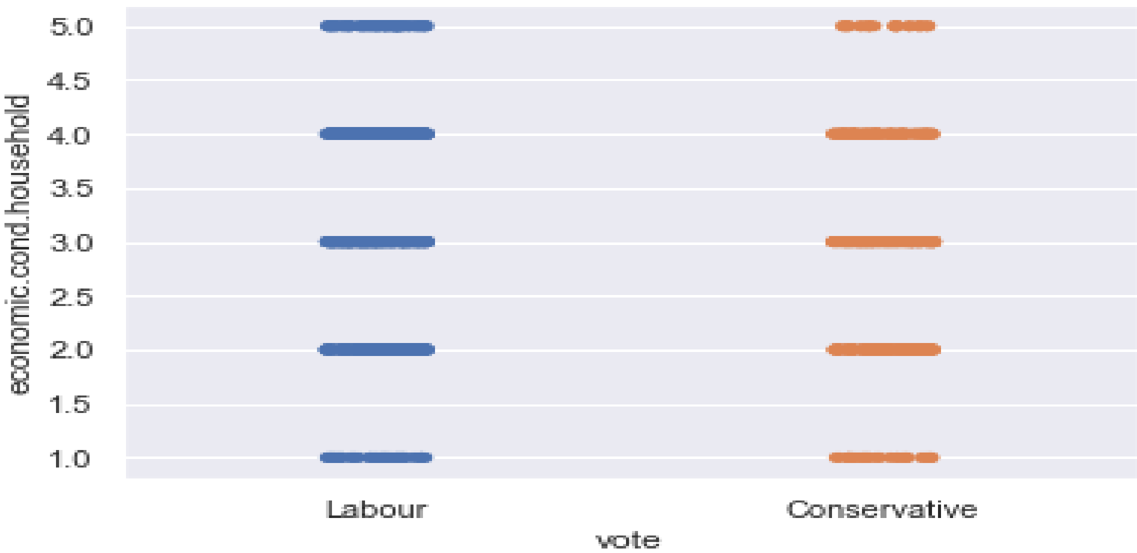

## Viewing the exact values of the variables of 'vote'with respect to 'economic.cond.national':

```
vote          economic.cond.national
Conservative  3                         199
              2                         140
              4                          91
              1                          21
              5                           9
Labour        4                         447
              3                         405
              2                         116
              5                          73
              1                          16
Name: economic.cond.national, dtype: int64
```

## Observation:

- Labour party has higher votes overall.
- Out of 82 people who gave a score of 5, 73 people have voted for the labour party.
- Out of 538 people who gave a score of 4, 447 people have voted for the labour party. This is the highest set of people in the labour party.
- Out of 604 people who gave a score of 3, 405 people have voted for the labour party. This is the 2nd highest set of people in the labour party. The remaining 199 people who have voted for the conservative party is the highest set of people in that party.
- Out of 256 people who gave a score of 2, 116 people have voted for the labour party. 140 people have voted for the conservative party. This is the instance where the conservative party has got more votes than the labour party.
- Out of 37 people who gave a score of 1, 16 people have voted for the labour party. 21 people have voted for the conservative party.
- The score of 3, 4 and 5 have more votes in the labour party.
- The score of 1 and 2 have more votes in the conservative party.

## Strip plot of 'vote' and 'economic.cond.household':




## Viewing the exact values of the variables of 'vote'with respect to 'economic.cond.household':

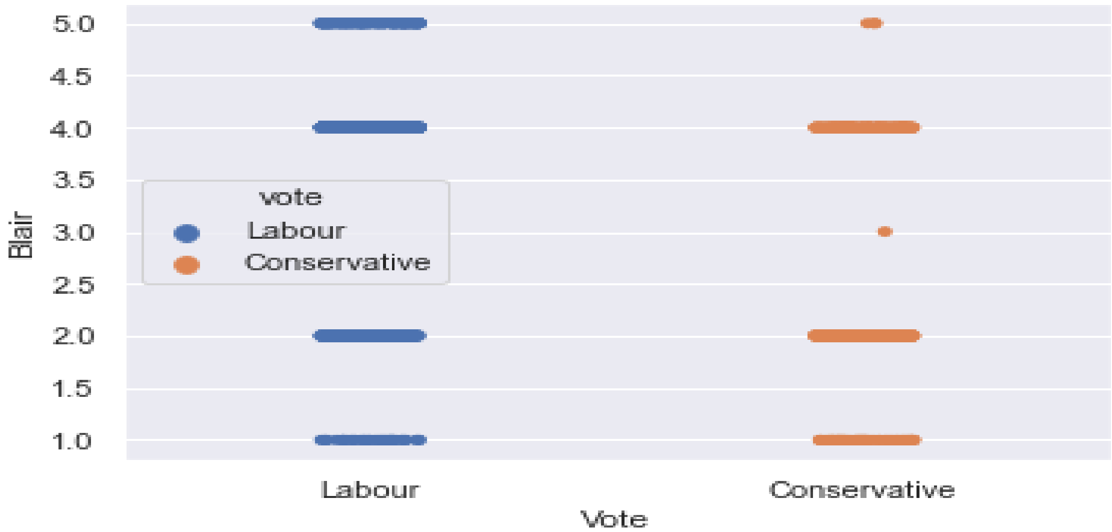
```
vote          economic.cond.household
Conservative  3                          197
              2                          126
              4                           86
              1                           28
              5                           23
Labour        3                          448
              4                          349
              2                          154
              5                           69
              1                           37
Name: economic.cond.household, dtype: int64
```

## Observation:

- Labour party has higher votes overall.
- Out of 92 people who gave a score of 5, 69 people have voted for the labour party.
- Out of 435 people who gave a score of 4, 349 people have voted for the labour party. This is the 2nd highest set of people in the labour party.

- Out of 645 people who gave a score of 3, 448 people have voted for the labour party. This is the highest set of people in the labour party. The remaining 197 people who have voted for the conservative party is the highest set of people in that party.
- Out of 280 people who gave a score of 2, 154 people have voted for the labour party. 126 people have voted for the conservative party.
- Out of 65 people who gave a score of 1, 37 people have voted for the labour party. 28 people have voted for the conservative party.
- In all the instances, the labour party have more votes than the conservative party.

## Strip plot of 'vote' and 'Blair':



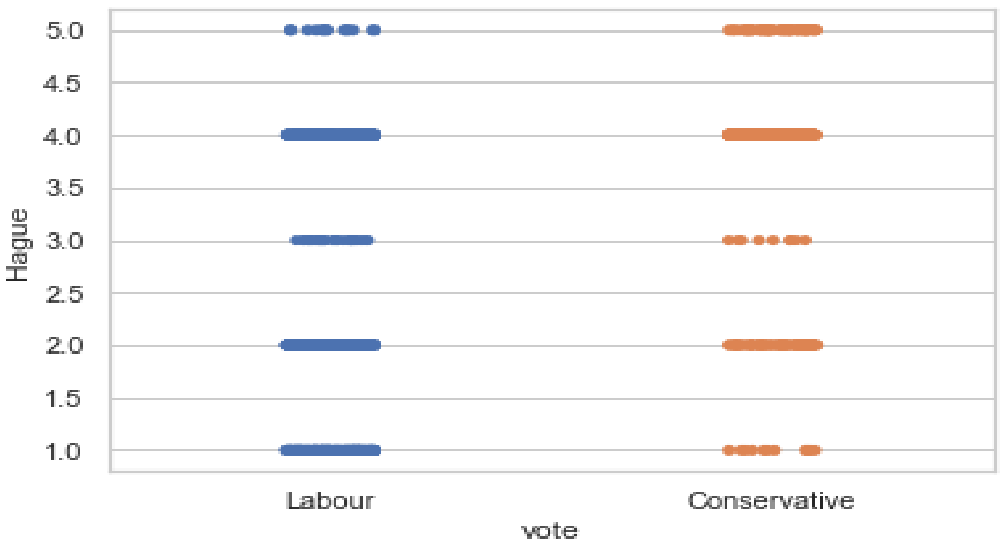

## Viewing the exact values of the variables of 'vote'with respect to 'Blair':

```
vote          Blair
Conservative  2        240
              4        157
              1         59
              5          3
              3          1
Labour        4        676
              2        194
              5        149
              1         38
Name: Blair, dtype: int64
```

## Observation:

- Labour party has higher votes overall.
- Out of 152 people who gave a score of 5, 149 people have voted for the labour party. The remaining 3 people, despite giving a score of 5 to the labour leader, have chosen to vote for the conservative party.
- Out of 833 people who gave a score of 4, 676 people have voted for the labour party. The remaining 157 people, despite giving a score of 4 to the labour leader, have chosen to vote for the conservative party.
- Only 1 person has given a score of 3 and that person hasvoted for the conservative party.
- Out of 434 people who gave a score of 2, 240 people have voted for the conservative party. The remaining 194 people, despite giving an unsatisfactory score of 2 to the labour leader, have chosen to vote for the labour party.
- Out of 97 people who gave a score of 1, 59 people have voted for the conservative party. The remaining 38 people, despite giving the lowest score of 1 to the labour leader, have chosen to vote for the labour party.
- The score of 4 and 5 have more votes in the labour party.
- The score of 1, 2 and 3 have more votes in the conservative party.

## Strip plot of 'vote' and 'Hague':




## Viewing the exact values of the variables of 'vote'with respect to 'Hague':

```
vote          Hague
Conservative  4        286
              2         95
              5         59
              1         11
              3          9
Labour        2        522
              4        271
              1        222
              3         28
              5         14
Name: Hague, dtype: int64
```

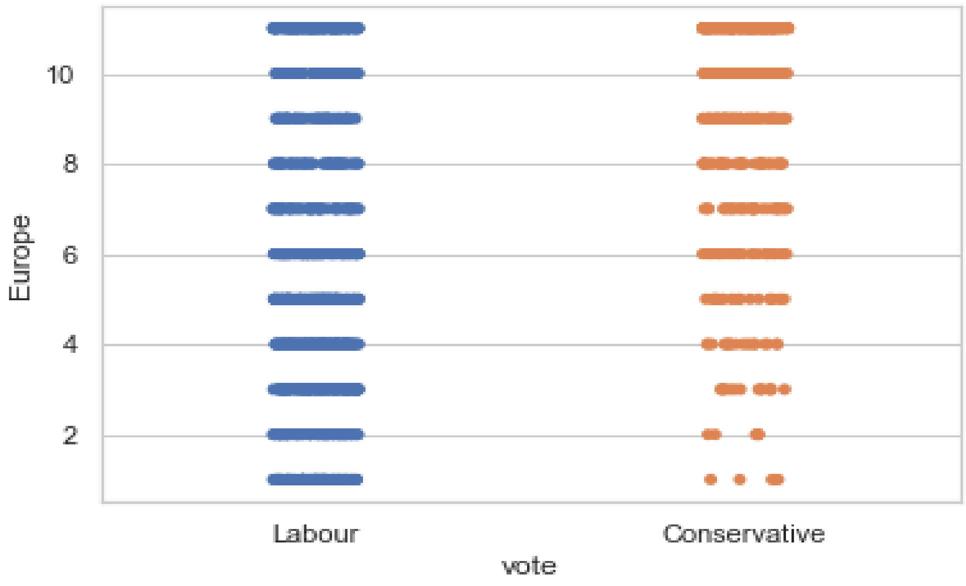
## Observation:

- Labour party has higher votes overall.
- Out of 73 people who gave a score of 5, 59 people have voted for the conservative party. The remaining 14people, despite giving a score of 5 to the conservative leader, have chosen to vote for the labour party.
- Out of 557 people who gave a score of 4, 286 people have voted for the conservative party. The remaining 271

people, despite giving a score of 4 to the conservative leader, have chosen to vote for the labour party.

- Out of 37 people who gave a score of 3, 28 have voted for the labour party. The remaining 9, despite giving an average score of 3 to the conservative party, have chosen to vote for the conservative party.
- Out of 617 people who gave a score of 2, 522 people have voted for the labour party. The remaining 95 people, despite giving an unsatisfactory score of 2 to the conservative leader, have chosen to vote for the conservative party.
- Out of 233 people who gave a score of 1, 222 people have voted for the labour party. The remaining 11 people, despite giving the lowest score of 1 to the conservative leader, have chosen to vote for the conservative party.
- The score of 4 and 5 have more votes in the conservative party, although in 4, the votes are almost equal in both the parties. Conservative party gets slightly higher.
- The score of 1, 2 and 3 have more votes in the labour party. Still, a significant percentage of people who gave a bad score to the conservative leader still chose to vote for 'Hague'.

## Strip plot of 'vote' and 'Europe':



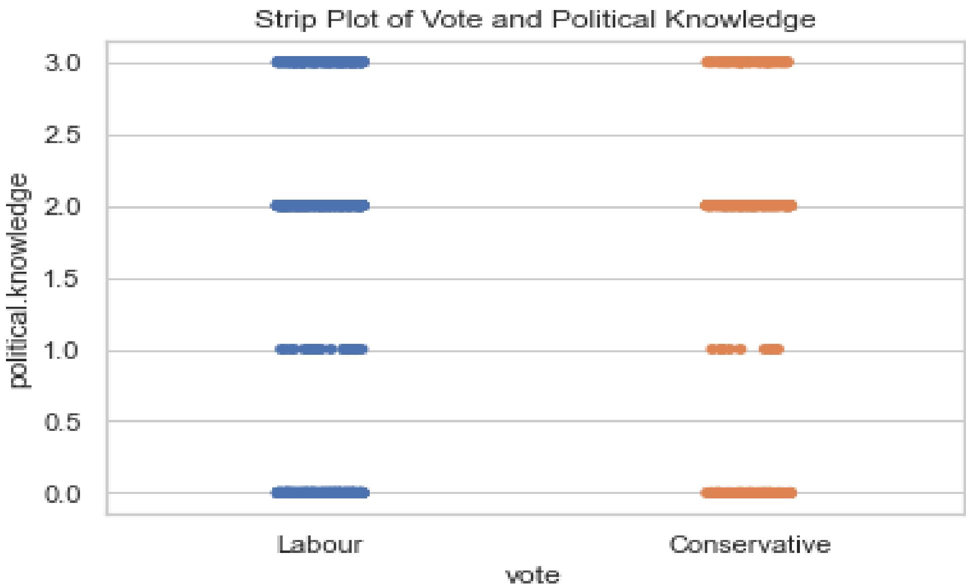

## Viewing the exact values of the variables of 'vote'with respect to 'Europe':

```
vote          Europe
Conservative  11        172
              9          56
              10         54
              8          48
              6          35
              7          32
              5          20
              4          18
              3          14
              2           6
              1           5
Labour        6         172
              11        166
              3         114
              4         108
              1         104
              5         103
              2          71
              8          63
              9          55
              7          54
              10         47
Name: Europe, dtype: int64
```

## Observation:

- Out of 338 people who gave a score of 11, 166 people have voted for the labour party and 172 people have voted for the conservative party.
- People who gave score of 7 to 10 have voted for labour and conservative almost equally. Conservative party seem to be slightly higher in these instances.
- Out of 207 people who gave a score of 6, 172 people have voted for the labour party and 35 people have voted for the conservative party.
- People who gave a score of 1 to 6 have predominantly voted for the labour party. As we can see, there are a total of 770 people who have given scores from 1 to 6. Out of 770 people, 672 people have voted for the labour party. So, 87.28% of the people have chosen labour party.
- So, we can infer that lower the 'Eurosceptic' sentiment, higher the votes for labour party.

## Strip plot of 'vote' and 'political.knowledge':



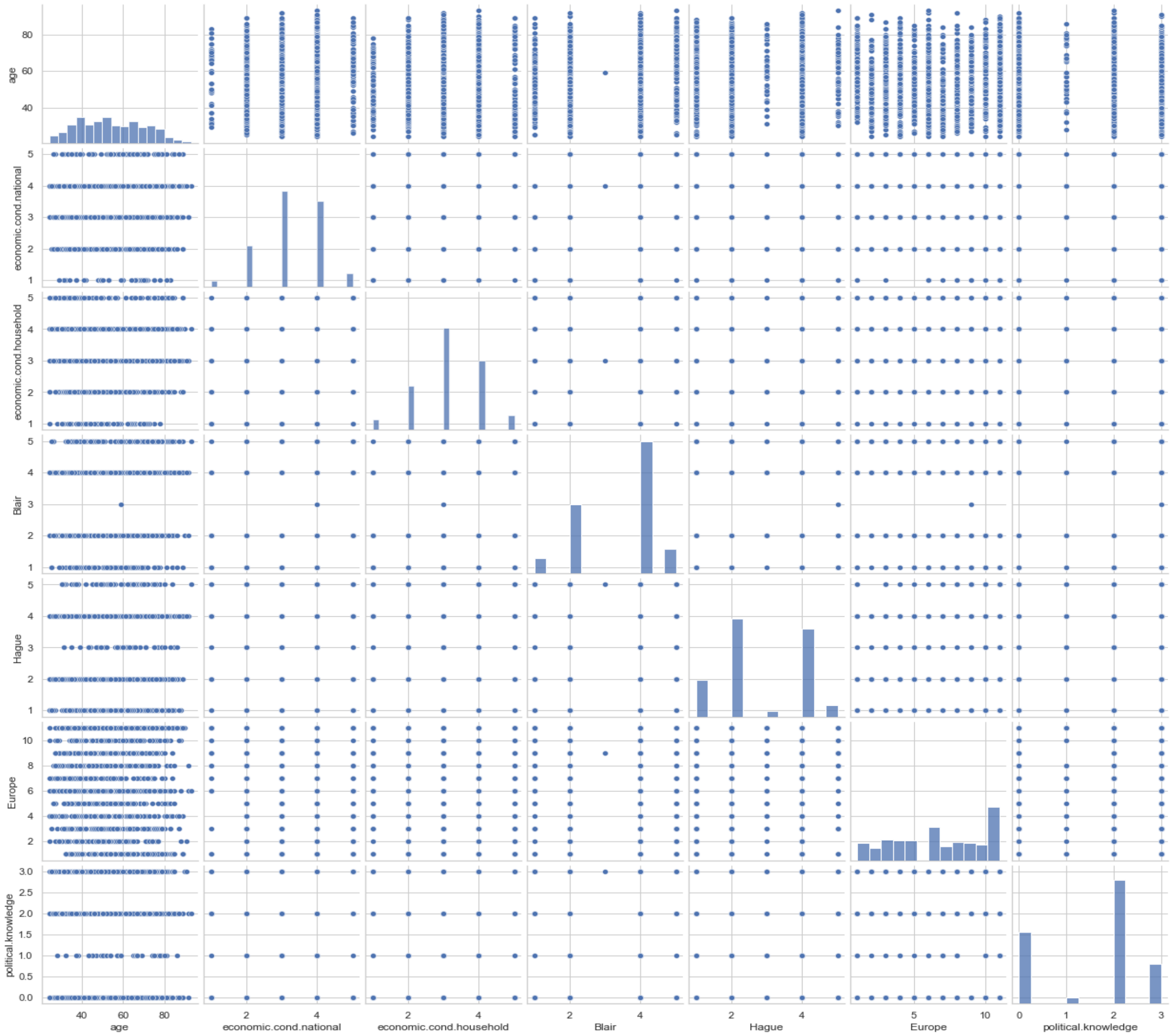

## Viewing the exact values of the variables of 'vote'with respect to 'political.knowledge':

```
vote          political.knowledge
Conservative  2                      283
              0                       94
              3                       72
              1                       11
Labour        2                      493
              0                      360
              3                      177
              1                       27
Name: political.knowledge, dtype: int64
```

## Observation:

- Out of 249 people who gave a score of 3, 177 people have voted for the labour party and 72 people have voted for the conservative party.
- Out of 776 people who gave a score of 2, 493 people have voted for the labour party and 283 people have voted for the conservative party.
- Out of 38 people who gave a score of 1, 27 people have voted for the labour party and 11 people have voted for the conservative party.
- Out of 454 people who gave a score of 0, 360 people have voted for the labour party and 94 people have voted for the conservative party.
- We can see that, in all instances, labour party gets the higher number of votes.
- Out of 1517 people, 454 people gave a score of 0. So, this means that, 29.93% of the people are casting their votes without any political knowledge.

## Checking pair-wise distribution of the continuous variables:




## Observation:

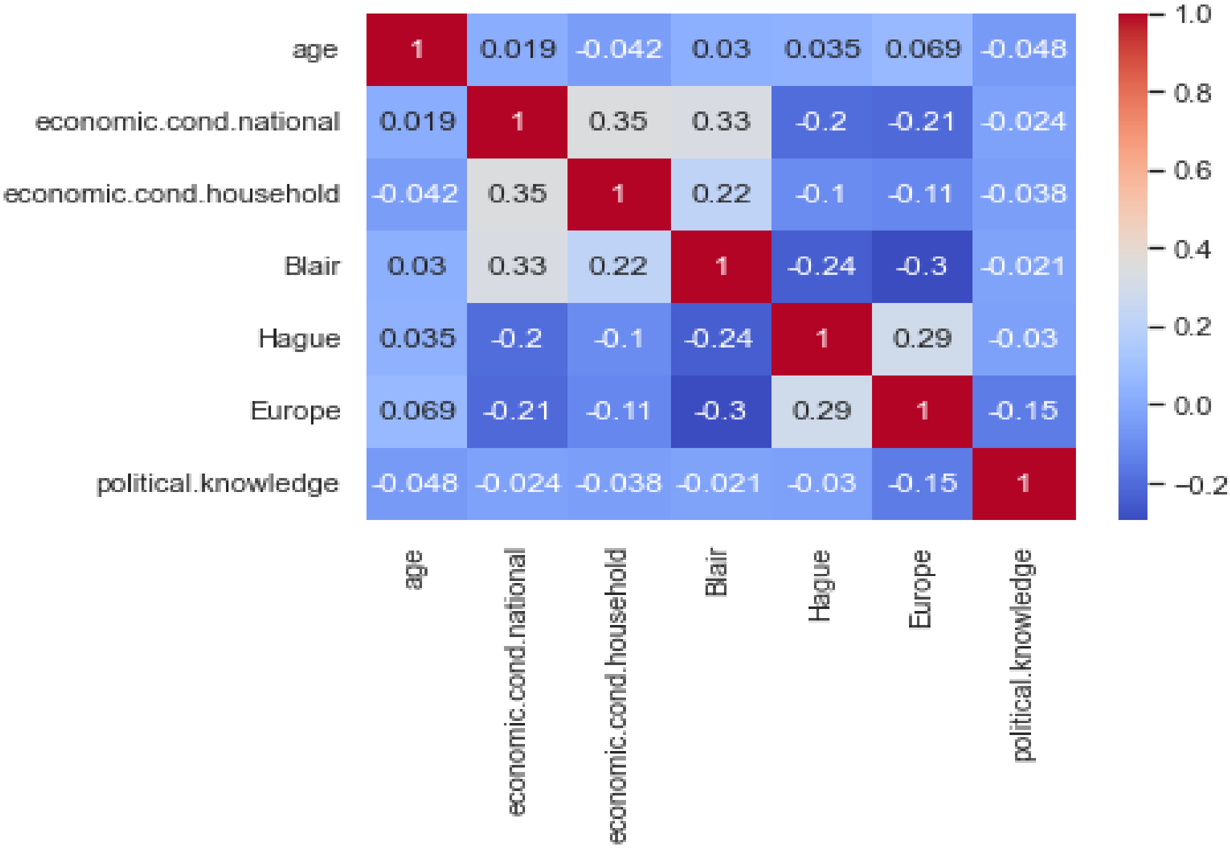
- Pair plot is a combination of histograms and scatter plots.
- From the histogram, we can see that, the 'Blair','Europe' and 'political.knowledge' variables are slightly left skewed.
- All other variables seem to be normally distributed.
- From the scatter plots, we can see that, there is mostly no correlation between the variables.

- We can use the correlation matrix to view them moreclearly.

Correlation matrix is a table which shows the correlation coefficient between variables. Correlation values range from -1 to +1. For values closer to zero, it means that, there is no linear trend between two variables. Values close to 1 means that the correlation is positive.

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

The correlation heat map helps us to visualize the correlationbetween two variables.



## Observation:

- We can see that, mostly there is no correlation in the dataset through this matrix. There are some variables that are moderately positively correlated and some that are slightly negatively correlated.
- ‘economic.cond.national’ with ‘economic.cond.household’ have moderate positive correlation.
- ‘Blair' with 'economic.cond.national' and ‘economic.cond.household’ have moderate positive correlation.
- ‘Europe’ with ‘Hague’ have moderate positive correlation.
- 'Hague' with 'economic.cond.national' and 'Blair' have moderate negative correlation.
- 'Europe' with 'economic.cond.national' and 'Blair' have moderate negative correlation.
- It's better to calculate the correlation matrix instead of the covariance matrix to understand the relationship between continuous variables.

| | age | economic.cond.national \ |
|---|---|---|
| age | 1.000000 | 0.018567 |
| economic.cond.national | 0.018567 | 1.000000 |
| economic.cond.household | -0.041587 | 0.346303 |
| Blair | 0.030218 | 0.326878 |
| Hague | 0.034626 | -0.199766 |
| Europe | 0.068880 | -0.209429 |
| political.knowledge | -0.048490 | -0.023624 |

| | economic.cond.household | Blair | Hague \ |
|---|---|---|---|
| age | -0.041587 | 0.030218 | 0.034626 |
| economic.cond.national | 0.346303 | 0.326878 | -0.199766 |
| economic.cond.household | 1.000000 | 0.215273 | -0.101956 |
| Blair | 0.215273 | 1.000000 | -0.243210 |
| Hague | -0.101956 | -0.243210 | 1.000000 |
| Europe | -0.114885 | -0.296162 | 0.287350 |
| political.knowledge | -0.037810 | -0.020917 | -0.030354 |

| | Europe | political.knowledge |
|---|---|---|
| age | 0.068880 | -0.048490 |
| economic.cond.national | -0.209429 | -0.023624 |
| economic.cond.household | -0.114885 | -0.037810 |
| Blair | -0.296162 | -0.020917 |
| Hague | 0.287350 | -0.030354 |
| Europe | 1.000000 | -0.152364 |
| political.knowledge | -0.152364 | 1.000000 |

**1.3) 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.**

## Viewing the data after encoding:

| | 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 |

## Encoded data info:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1517 entries, 0 to 1524
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   age                      1517 non-null   int64
 1   economic.cond.national   1517 non-null   int64
 2   economic.cond.household  1517 non-null   int64
 3   Blair                    1517 non-null   int64
 4   Hague                    1517 non-null   int64
 5   Europe                   1517 non-null   int64
 6   political.knowledge      1517 non-null   int64
 7   vote_Labour              1517 non-null   uint8
 8   gender_male              1517 non-null   uint8
dtypes: int64(7), uint8(2)
memory usage: 130.1 KB
```

## Train-test-split:

Our model will use all the variables and 'vote_Labour' is the target variable. The train-test split is a technique for evaluating the performance of a machine learning algorithm. The procedure involves taking a dataset and dividing it into two subsets.

- **Train Dataset:** Used to fit the machine learning model.
- **Test Dataset:** Used to evaluate the fit machine learning model.

The data is divided into 2 subsets, training and testing set. Earlier, we have extracted the target variable 'vote_Labour' ina separate vector for subsets. Random state chosen as 1.

- **Training Set:** 70percent of data.
- **Testing Set:** 30 percent of the data.

## Train-Test-Split Shape:

```
x_train: (1061, 8)
y_train: (1061, 1)
x_test: (456, 8)
y_test: (456, 1)
```

## Why scaling?:

- The dataset contains features highly varying in magnitudes, units and range between the 'age' column and other columns.
- But since, most of the machine learning algorithms use Eucledian distance between two data points in their computations, this is a problem.
- If left alone, these algorithms only take in the magnitude of features neglecting the units.
- The results would vary greatly between different units, 1km and 1000 meters.
- The features with high magnitudes will weigh in a lot more in the distance calculations than features with low magnitudes.
- To suppress this effect, we need to bring all features to the same level of magnitudes. This can be achieved by scaling.
- In this case, we have a lot of encoded, ordinal, categorical and continuous variables. So, we use the min max scalar technique to scale the data.

## Viewing the data after scaling:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.275362 | 0.50 | 0.50 | 0.75 | 0.00 | 0.1 | 0.666667 | 0.0 |
| 1 | 0.173913 | 0.75 | 0.75 | 0.75 | 0.75 | 0.4 | 0.666667 | 1.0 |
| 2 | 0.159420 | 0.75 | 0.75 | 1.00 | 0.25 | 0.2 | 0.666667 | 1.0 |
| 3 | 0.000000 | 0.75 | 0.25 | 0.25 | 0.00 | 0.3 | 0.000000 | 0.0 |
| 4 | 0.246377 | 0.25 | 0.25 | 0.00 | 0.00 | 0.5 | 0.666667 | 1.0 |

**1.4) Apply Logistic Regression and LDA (Linear Discriminant Analysis) (2 pts). Interpret the inferences of both model s (2 pts). Successful implementation of each model. Logical reason should be shared if any custom changes are made to the parameters while building the model. Calculate Train and Test Accuracies for each model. Comment on the validness of models (over fitting or under fitting)**

## Logistic Regression Model:

There are no outliers present in the continuous variable 'age'. The remaining variables are categorical in nature. Our model will use all the variables and 'vote_Labour' is the target variable.

## Accuracy - Train data:

```
0.8341187558906692
```

## Accuracy - Test data:

```
0.8267543859649122
```

## Classification report - Train data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.76 | 0.63 | 0.69 | 307 |
| 1 | 0.86 | 0.92 | 0.89 | 754 |
| accuracy | | | 0.83 | 1061 |
| macro avg | 0.81 | 0.77 | 0.79 | 1061 |
| weighted avg | 0.83 | 0.83 | 0.83 | 1061 |

## Classification report - Test data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.76 | 0.71 | 0.73 | 153 |
| 1 | 0.86 | 0.89 | 0.87 | 303 |
| accuracy | | | 0.83 | 456 |
| macro avg | 0.81 | 0.80 | 0.80 | 456 |
| weighted avg | 0.82 | 0.83 | 0.83 | 456 |

## Logistic Regression Model - ObservationTrain data:

- Accuracy: 83.41%
- Precision: 86%
- Recall: 92%
- F1-Score: 89%

## Test data:

- Accuracy: 82.68%
- Precision: 86%
- Recall: 89%
- F1-Score: 87%

## Validness of the model:

- The model is not over-fitted or under-fitted.
- The error in the test data is slightly higher than the train data, which is absolutely fine because the error margin is low and the error in both train and test data is not too high. Thus, the model is not over-fitted or under-fitted.

## Linear Discriminant Analysis Model:

There are no outliers present in the continuous variable 'age'. The remaining variables are categorical in nature. Our model will use all the variables and 'vote_Labour' is the target variable.

## Accuracy - Train data:

```
0.8341187558906692
```

## Accuracy - Test data:

```
0.8333333333333334
```

## Classification report - Train data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.74 | 0.65 | 0.69 | 307 |
| 1 | 0.86 | 0.91 | 0.89 | 754 |
| accuracy | | | 0.83 | 1061 |
| macro avg | 0.80 | 0.78 | 0.79 | 1061 |
| weighted avg | 0.83 | 0.83 | 0.83 | 1061 |

## Classification report - Test data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.77 | 0.73 | 0.74 | 153 |
| 1 | 0.86 | 0.89 | 0.88 | 303 |
| accuracy | | | 0.83 | 456 |
| macro avg | 0.82 | 0.81 | 0.81 | 456 |
| weighted avg | 0.83 | 0.83 | 0.83 | 456 |

## Linear Discriminant Analysis Model - Observation

## Train data:

- Accuracy: 83.41%

- Precision: 86%
- Recall: 91%
- F1-Score: 89%

## Test data:

- Accuracy: 83.33%
- Precision: 86%
- Recall: 89%
- F1-Score: 88%

## Validness of the model:

- The model is not over-fitted or under-fitted.
- The error in the test data is slightly higher than the train data, which is absolutely fine because the error margin is low and the error in both train and test data is not too high. Thus, the model is not over-fitted or under-fitted.

**1.5) Apply KNN Model and Naïve Bayes Model (2pts). Interpret the inferences of each model (2 pts). Successful implementation of each model. Logical reason should be shared if any custom changes are made to the parameters while building the model. Calculate Train and Test Accuracies for each model. Comment on the validness of models (over fitting or under fitting)**

## K-Nearest Neighbor Model:

There are no outliers present in the continuous variable 'age'. The remaining variables are categorical in nature. Our model will use all the variables and 'vote_Labour' is the target variable. We take K value as 7.

## Accuracy - Train data:

```
1.0
```

## Accuracy - Test data:

```
0.8377192982456141
```

## Classification report - Train data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 307 |
| 1 | 1.00 | 1.00 | 1.00 | 754 |
| accuracy | | | 1.00 | 1061 |
| macro avg | 1.00 | 1.00 | 1.00 | 1061 |
| weighted avg | 1.00 | 1.00 | 1.00 | 1061 |

## Classification report - Test data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.78 | 0.71 | 0.75 | 153 |
| 1 | 0.86 | 0.90 | 0.88 | 303 |
| accuracy | | | 0.84 | 456 |
| macro avg | 0.82 | 0.81 | 0.81 | 456 |
| weighted avg | 0.84 | 0.84 | 0.84 | 456 |

## K-Nearest Neighbor Model - ObservationTrain data:

- Accuracy: 100%
- Precision: 100%
- Recall: 100%
- F1-Score: 100%

## Test data:

- Accuracy: 83.77%
- Precision: 86%
- Recall: 90%
- F1-Score: 88%

## Validness of the model:

- The model is over-fitted.
- As we can see, the train data has a 100% accuracy andtest data has 84% accuracy. The difference is more than 10%. So, we can infer that the KNN model is over-fitted.

## Naïve Bayes Model:

There are no outliers present in the continuous variable 'age'. The remaining variables are categorical in nature. Our model will use all the variables and 'vote_Labour' is the target variable.

## Accuracy - Train data:

```
0.8350612629594723
```

## Accuracy - Test data:

```
0.8223684210526315
```

## Classification report - Train data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.73 | 0.69 | 0.71 | 307 |
| 1 | 0.88 | 0.90 | 0.89 | 754 |
| accuracy | | | 0.84 | 1061 |
| macro avg | 0.80 | 0.79 | 0.80 | 1061 |
| weighted avg | 0.83 | 0.84 | 0.83 | 1061 |

## Classification report - Test data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.74 | 0.73 | 0.73 | 153 |
| 1 | 0.87 | 0.87 | 0.87 | 303 |
| accuracy | | | 0.82 | 456 |
| macro avg | 0.80 | 0.80 | 0.80 | 456 |
| weighted avg | 0.82 | 0.82 | 0.82 | 456 |

## Naïve Bayes Model - Observation

### Train data:

- Accuracy: 83.51%
- Precision: 88%
- Recall: 90%
- F1-Score: 89%

### Test data:

- Accuracy: 82.24%
- Precision: 87%
- Recall: 87%
- F1-Score: 87%

### Validness of the model:

- The model is not over-fitted or under-fitted.
- The error in the test data is slightly higher than the train data, which is absolutely fine because the error margin is low and the error in both train and test data is not too high. Thus, the model is not over-fitted or under-fitted.

**1.6) Model Tuning (4 pts) , Bagging ( 1.5 pts) and Boosting (1.5 pts). Apply grid search on each model (include all models) and make models on best_params. 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.**

## Logistic Regression Model Tuning:

## Best parameters:

```
{'C': 0.615848211066026,
 'max_iter': 100,
 'penalty': 'l1',
 'solver': 'liblinear',
 'tol': 0.0001}
```

## Accuracy - Train data:

```
0.8360037700282752
```

## Accuracy - Test data:

```
0.8421052631578947
```

## Classification report - Train data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.76 | 0.64 | 0.69 | 307 |
| 1 | 0.86 | 0.92 | 0.89 | 754 |
| accuracy | | | 0.84 | 1061 |
| macro avg | 0.81 | 0.78 | 0.79 | 1061 |
| weighted avg | 0.83 | 0.84 | 0.83 | 1061 |

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.79 | 0.72 | 0.75 | 153 |
| 1 | 0.86 | 0.90 | 0.88 | 303 |
| accuracy | | | 0.84 | 456 |
| macro avg | 0.83 | 0.81 | 0.82 | 456 |
| weighted avg | 0.84 | 0.84 | 0.84 | 456 |

## Classification report - Test data.

## Logistic Regression Model Tuned - Observation

## Train data:

- Accuracy: 83.6%
- Precision: 86%
- Recall: 92%
- F1-Score: 89%

## Test data:

- Accuracy: 84.21%
- Precision: 86%
- Recall: 90%
- F1-Score: 88%

## Comparison on performance of both regular and tuned logistic regression models:

| | Regular Model (%) | Tuned Model (%) |
|---|---|---|
| **Train:** | | |
| **Accuracy** | 83.41 | 83.6 |
| **Precision** | 86 | 86 |
| **Recall** | 92 | 92 |
| **F1-score** | 89 | 89 |
| **Test:** | | |
| **Accuracy** | 82.68 | 84.21 |
| **Precision** | 86 | 86 |
| **Recall** | 89 | 90 |
| **F1-score** | 87 | 88 |

- As we can see from the above tabular comparison, there is not much difference between the performance regular LR model and tuned LR model.
- The values are high overall and there is no over-fitting or under-fitting. Therefore both models are equally good models.

## Linear Discriminant Analysis Model Tuning: Best parameters:

```
{'solver': 'svd', 'tol': 0.0001}
```

## Accuracy - Train data:

```
0.8322337417530632
```

## Accuracy - Test data:

```
0.8399122807017544
```

## Classification report - Train data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.74 | 0.65 | 0.69 | 307 |
| 1 | 0.87 | 0.90 | 0.88 | 754 |
| accuracy | | | 0.83 | 1061 |
| macro avg | 0.80 | 0.78 | 0.79 | 1061 |
| weighted avg | 0.83 | 0.83 | 0.83 | 1061 |

## Classification report - Test data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.77 | 0.74 | 0.76 | 153 |
| 1 | 0.87 | 0.89 | 0.88 | 303 |
| accuracy | | | 0.84 | 456 |
| macro avg | 0.82 | 0.81 | 0.82 | 456 |
| weighted avg | 0.84 | 0.84 | 0.84 | 456 |

## LDA Model Tuned - Observation

## Train data:

- Accuracy: 83.22%
- Precision: 87%
- Recall: 90%
- F1-Score: 88%

## Test data:

- Accuracy: 83.99%
- Precision: 87%
- Recall: 89%
- F1-Score: 88%

## Comparison on performance of both regular and tuned LDA models:

| | Regular Model (%) | Tuned Model (%) |
|---|---|---|
| **Train:** | | |
| **Accuracy** | 83.41 | 83.22 |
| **Precision** | 86 | 87 |
| **Recall** | 91 | 90 |
| **F1-score** | 89 | 88 |
| **Test:** | | |
| **Accuracy** | 83.33 | 83.99 |
| **Precision** | 86 | 87 |
| **Recall** | 89 | 89 |
| **F1-score** | 88 | 88 |

- As we can see from the above tabular comparison, thereis not much difference between the performance of regular LDA model and tuned LDA model.
- The values are high overall and there is no over-fitting or under-fitting. Therefore both models are equally good models.

## K-Nearest Neighbour Model Tuning:

## Best parameters:

```
{'leaf_size': 15, 'n_neighbors': 21, 'weights': 'uniform'}
```

## Accuracy - Train data:

```
0.8435438265786993
```

## Accuracy - Test data:

```
0.8618421052631579
```

## Classification report - Train data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.75 | 0.69 | 0.72 | 307 |
| 1 | 0.88 | 0.91 | 0.89 | 754 |
| accuracy | | | 0.84 | 1061 |
| macro avg | 0.81 | 0.80 | 0.80 | 1061 |
| weighted avg | 0.84 | 0.84 | 0.84 | 1061 |

## Classification report - Test data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.84 | 0.73 | 0.78 | 153 |
| 1 | 0.87 | 0.93 | 0.90 | 303 |
| accuracy | | | 0.86 | 456 |
| macro avg | 0.85 | 0.83 | 0.84 | 456 |
| weighted avg | 0.86 | 0.86 | 0.86 | 456 |

## KNN Model Tuned - Observation

## Train data:

- Accuracy: 84.35%
- Precision: 88%
- Recall: 91%
- F1-Score: 89%

## Test data:

- Accuracy: 86.18%
- Precision: 87%
- Recall: 93%
- F1-Score: 90%

## Comparison on performance of both regular and tuned KNN models:

| | Regular Model (%) | Tuned Model (%) |
|---|---|---|
| **Train:** | | |
| **Accuracy** | 100 | 84.35 |
| **Precision** | 100 | 88 |
| **Recall** | 100 | 91 |
| **F1-score** | 100 | 89 |
| **Test:** | | |
| **Accuracy** | 83.77 | 86.18 |
| **Precision** | 86 | 87 |
| **Recall** | 90 | 93 |
| **F1-score** | 88 | 90 |

- There is no over-fitting or under-fitting in the tuned KNN model. Overall, it is a good model.
- As we can see, the regular KNN model was over-fitted. But model tuning has helped the model to recover from over-fitting.
- The values are better in the tuned KNN model.
- Therefore, the tuned KNN model is a better model.

## Ensemble Random Forest Classifier Feature importances:

```
        Imp
0  0.215348
5  0.185293
4  0.181437
3  0.136335
1  0.094462
2  0.077034
6  0.075300
7  0.034792
```

- Here,
- 0 = age
- 1 = economic.cond.national
- 2 = economic.cond.household
- 3 = Blair
- 4 = Hague
- 5 = Europe
- 6 = political.knowledge
- 7 = gender_male

## Accuracy - Train data:

```
1.0
```

## Accuracy - Test data:

```
0.8267543859649122
```

## Classification report - Train data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 307 |
| 1 | 1.00 | 1.00 | 1.00 | 754 |
| accuracy | | | 1.00 | 1061 |
| macro avg | 1.00 | 1.00 | 1.00 | 1061 |
| weighted avg | 1.00 | 1.00 | 1.00 | 1061 |

## Classification report - Test data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.79 | 0.65 | 0.72 | 153 |
| 1 | 0.84 | 0.91 | 0.88 | 303 |
| accuracy | | | 0.83 | 456 |
| macro avg | 0.82 | 0.78 | 0.80 | 456 |
| weighted avg | 0.82 | 0.83 | 0.82 | 456 |

## Random Forest Classifier - Observation

## Train data:

- Accuracy: 100%
- Precision: 100%
- Recall: 100%
- F1-Score: 100%

## Test data:

- Accuracy: 82.68%
- Precision: 84%
- Recall: 91%
- F1-Score: 88%

The model is over-fitted. We will use bagging to improve the performance of the model.

## Ensemble technique - Bagging

## Accuracy - Train data:

```
0.9538171536286523
```

## Accuracy - Test data:

```
0.8245614035087719
```

The RF model even after using bagging technique, is still over-fitted.

## Ensemble technique - AdaBoosting

## Accuracy - Train data:

```
0.8426013195098964
```

## Accuracy - Test data:

```
0.8201754385964912
```

The model is not over-fitted. The values are good. Therefore,the model is a good model.

## Ensemble technique - Gradient Boosting

## Accuracy - Train data:

```
0.8925541941564562
```

## Accuracy - Test data:

```
0.8333333333333334
```

The model is not over-fitted. The values are better than AdaBoosting model. The model is a good model.

## Random Forest model Tuning

## Best parameters:

```
{'criterion': 'gini',
 'max_depth': 8,
 'max_features': 5,
 'min_samples_leaf': 9,
 'min_samples_split': 50,
 'n_estimators': 100,
 'random_state': 1}
```

## Bagging tuned:

## Accuracy - Train data:

```
0.8444863336475024
```

## Accuracy - Test data:

```
0.8135964912280702
```

## Classification report - Train data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.81 | 0.61 | 0.69 | 307 |
| 1 | 0.86 | 0.94 | 0.90 | 754 |
| accuracy | | | 0.84 | 1061 |
| macro avg | 0.83 | 0.77 | 0.79 | 1061 |
| weighted avg | 0.84 | 0.84 | 0.84 | 1061 |

## Classification report - Test data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.79 | 0.61 | 0.69 | 153 |
| 1 | 0.82 | 0.92 | 0.87 | 303 |
| accuracy | | | 0.81 | 456 |
| macro avg | 0.81 | 0.76 | 0.78 | 456 |
| weighted avg | 0.81 | 0.81 | 0.81 | 456 |

The tuning of the model has help the model recover from over-fitting. Now the model is a good model.

## Random Forest tuned - Ada

## Boosting Accuracy - Train data:

```
0.9349670122525919
```

## Accuracy - Test data:

```
0.831140350877193
```

## Classification Report - Train data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 0.87 | 0.89 | 307 |
| 1 | 0.95 | 0.96 | 0.95 | 754 |
| accuracy | | | 0.93 | 1061 |
| macro avg | 0.92 | 0.92 | 0.92 | 1061 |
| weighted avg | 0.93 | 0.93 | 0.93 | 1061 |

## Classification Report - Test data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.77 | 0.71 | 0.74 | 153 |
| 1 | 0.86 | 0.89 | 0.88 | 303 |
| accuracy | | | 0.83 | 456 |
| macro avg | 0.81 | 0.80 | 0.81 | 456 |
| weighted avg | 0.83 | 0.83 | 0.83 | 456 |

There is no over-fitting. There is improvement from the regular model. The model is a good model.

## Gradient Boosting Classifier Tuned

## Best parameters:

```
{'criterion': 'friedman_mse',
 'loss': 'deviance',
 'max_depth': 8,
 'max_features': 'log2',
 'min_samples_leaf': 15,
 'n_estimators': 20,
 'subsample': 0.8}
```

## Accuracy - Train data:

```
0.883129123468426
```

## Accuracy - Test data:

```
0.8728070175438597
```

## Classification Report - Train data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.85 | 0.73 | 0.78 | 307 |
| 1 | 0.89 | 0.95 | 0.92 | 754 |
| accuracy | | | 0.88 | 1061 |
| macro avg | 0.87 | 0.84 | 0.85 | 1061 |
| weighted avg | 0.88 | 0.88 | 0.88 | 1061 |

## Classification Report - Test data:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.86 | 0.75 | 0.80 | 153 |
| 1 | 0.88 | 0.94 | 0.91 | 303 |
| accuracy | | | 0.87 | 456 |
| macro avg | 0.87 | 0.84 | 0.85 | 456 |
| weighted avg | 0.87 | 0.87 | 0.87 | 456 |

- The gradient boost classifier after tuning, has improved the model significantly.
- The difference between the train and test accuracies has also been reduced.
- Overall, the tuned Gradient Boost classifier is a better model.

**1.7 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model, classification report (4 pts) Final Model - Compare and comment on all models on the basis of the performance metrics in a structured tabular manner. Describe on which model is best/optimized, After comparison which model suits the best for the problem in hand on the basis of different measures. Comment on the final model.(3 pts)**

## Logistic Regression Model - Regular:

## Predicted Class and probs:

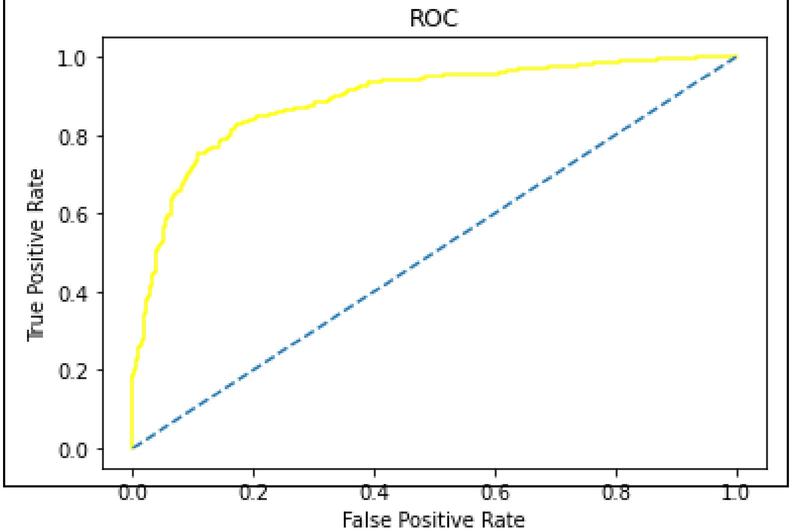
| | 0 | 1 |
|---|---|---|
| 0 | 0.417551 | 0.582449 |
| 1 | 0.167205 | 0.832795 |
| 2 | 0.010468 | 0.989532 |
| 3 | 0.799987 | 0.200013 |
| 4 | 0.087930 | 0.912070 |

## Accuracy - Train:

0.8341187558906692

## ROC and AUC - Train:

AUC: 0.8898383431686813

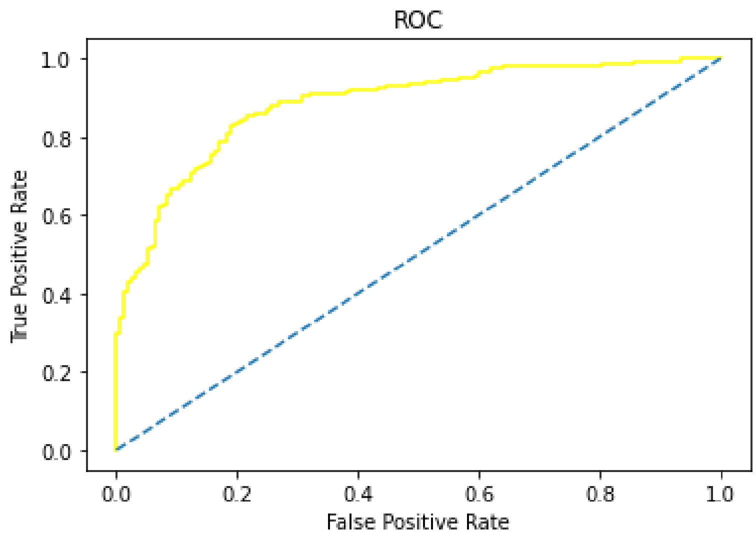


## Accuracy - Test:

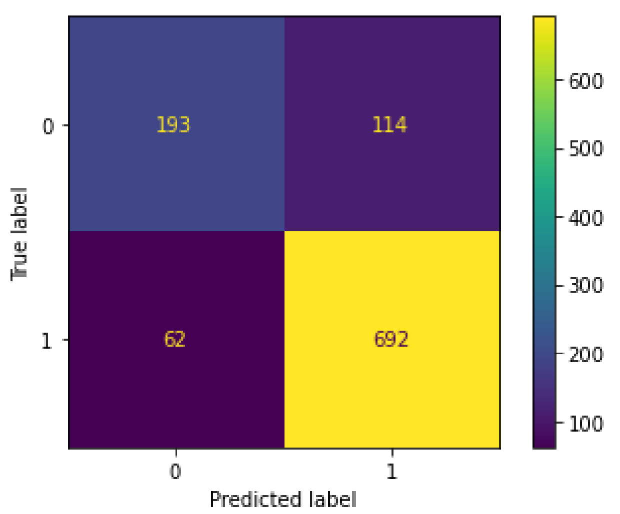
```
0.8267543859649122
```

## ROC and AUC - Test:

```
AUC: 0.8840786039388253
```



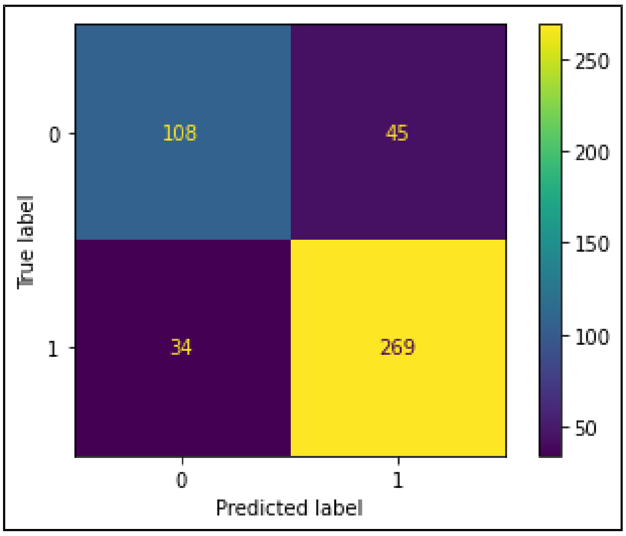

## Confusion matrix - Train:



## Classification report - Train:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.76 | 0.63 | 0.69 | 307 |
| 1 | 0.86 | 0.92 | 0.89 | 754 |
| accuracy | | | 0.83 | 1061 |
| macro avg | 0.81 | 0.77 | 0.79 | 1061 |
| weighted avg | 0.83 | 0.83 | 0.83 | 1061 |

## Confusion matrix - Test:




## Classification report - Test:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.76 | 0.71 | 0.73 | 153 |
| 1 | 0.86 | 0.89 | 0.87 | 303 |
| accuracy | | | 0.83 | 456 |
| macro avg | 0.81 | 0.80 | 0.80 | 456 |
| weighted avg | 0.82 | 0.83 | 0.83 | 456 |

## Observation:

## Train data:

- Accuracy: 83.41%
- Precision: 86%
- Recall: 92%
- F1-Score: 89%

- AUC: 88.98%

## Test data:

- Accuracy: 82.68%
- Precision: 86%
- Recall: 89%
- F1-Score: 87%
- AUC: 88.4%

The model is not over-fitted or under-fitted. It is a good model.

## Logistic Regression Model - Tuned:Predicted Class and probs:

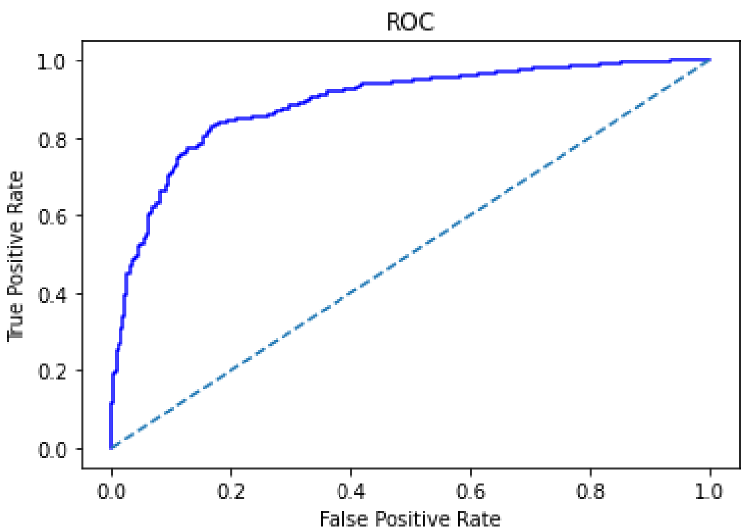
| | 0 | 1 |
|---|---|---|
| 0 | 0.429309 | 0.570691 |
| 1 | 0.172745 | 0.827255 |
| 2 | 0.014362 | 0.985638 |
| 3 | 0.793553 | 0.206447 |
| 4 | 0.105217 | 0.894783 |

## Accuracy - Train:

```
0.8360037700282752
```

## ROC and AUC - Train:

```
AUC: 0.8888533683546601
```

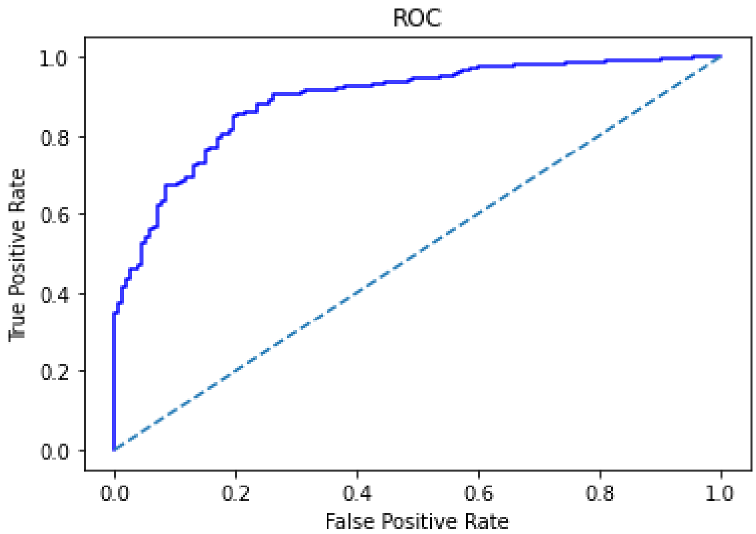


## Accuracy - Test:

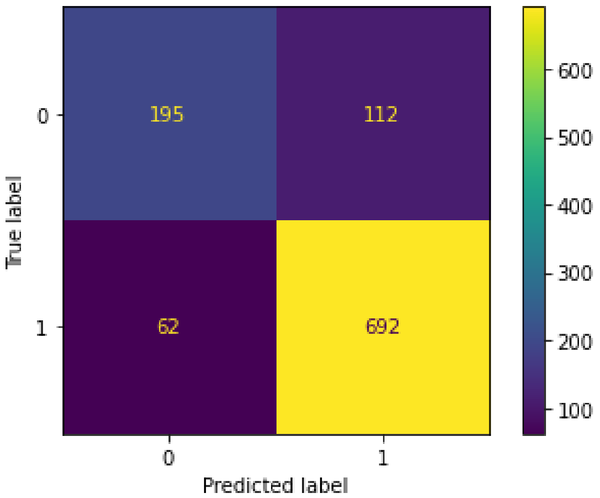
```
0.8421052631578947
```

## ROC and AUC - Test:

```
AUC: 0.8905066977285964
```



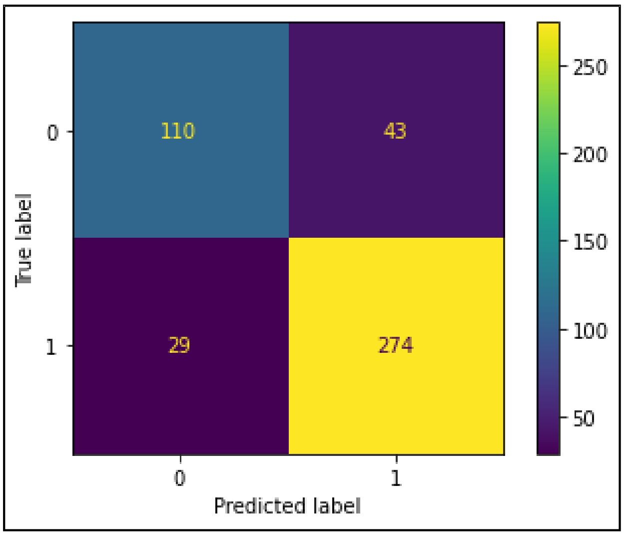

## Confusion matrix - Train:



## Classification report - Train:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.76 | 0.64 | 0.69 | 307 |
| 1 | 0.86 | 0.92 | 0.89 | 754 |
| accuracy | | | 0.84 | 1061 |
| macro avg | 0.81 | 0.78 | 0.79 | 1061 |
| weighted avg | 0.83 | 0.84 | 0.83 | 1061 |

## Confusion matrix - Test:




## Classification report - Test:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.79 | 0.72 | 0.75 | 153 |
| 1 | 0.86 | 0.90 | 0.88 | 303 |
| accuracy | | | 0.84 | 456 |
| macro avg | 0.83 | 0.81 | 0.82 | 456 |
| weighted avg | 0.84 | 0.84 | 0.84 | 456 |

## Observation:

## Train data:

- Accuracy: 83.6%
- Precision: 86%
- Recall: 92%
- F1-Score: 89%
- AUC: 88.89%

## Test data:

- Accuracy: 84.21%
- Precision: 86%
- Recall: 90%
- F1-Score: 88%
- AUC: 89.05%

## Comparison between the regular LR model and tuned LR model:

- As we can see, there is not much difference between the performance of regular LR model and tuned LR model.
- The values are high overall and there is no over-fitting or under- fitting. Therefore both models are equally good models.

## LDA Model - Regular: Predicted Class and probs:

| | 0 | 1 |
|---|---|---|
| 0 | 0.462093 | 0.537907 |
| 1 | 0.133955 | 0.866045 |
| 2 | 0.006414 | 0.993586 |
| 3 | 0.861210 | 0.138790 |
| 4 | 0.056545 | 0.943455 |

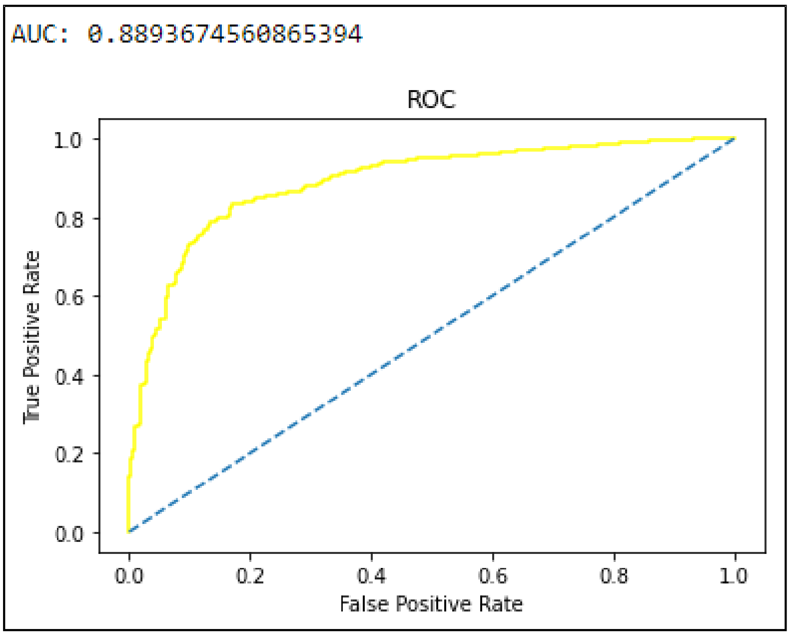
## Accuracy - Train:

```
0.8341187558906692
```

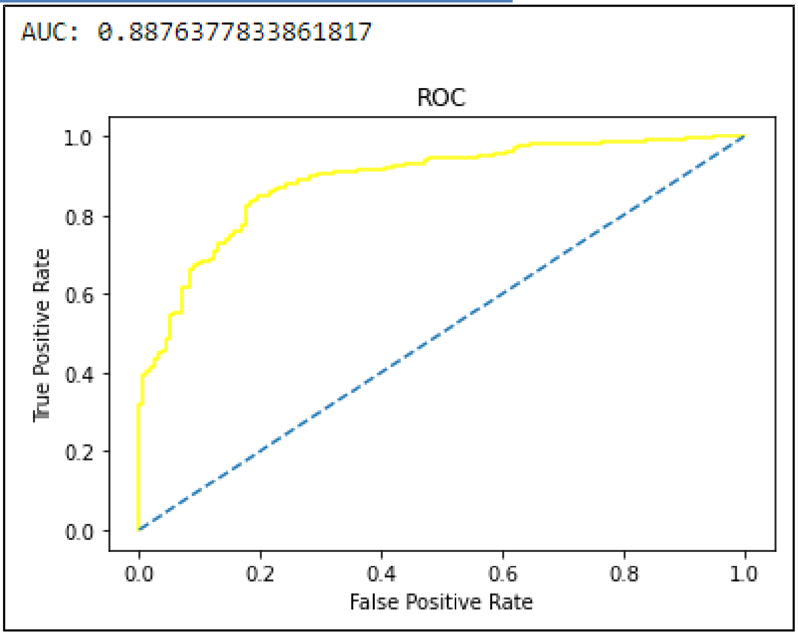
## Accuracy - Test:

```
0.8333333333333334
```

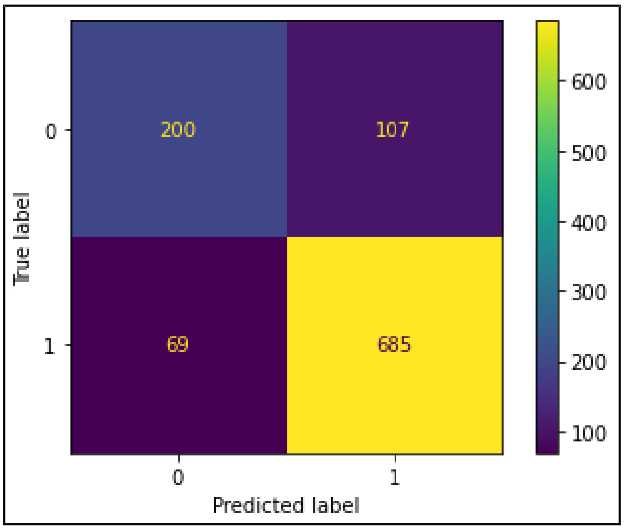
## ROC and AUC - Train:




## ROC and AUC - Test:



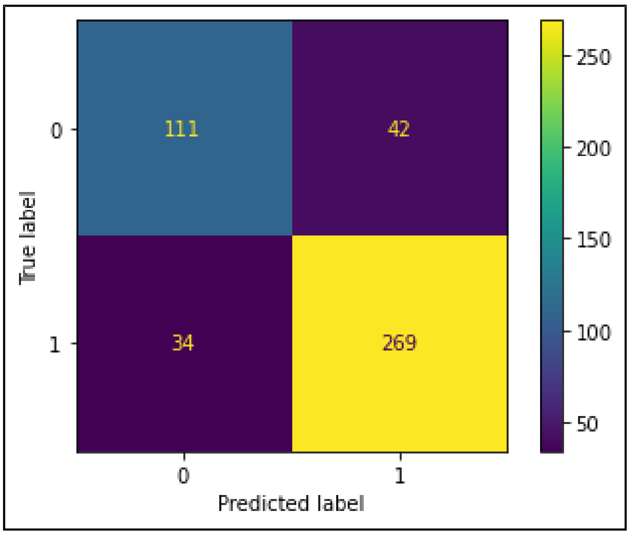

## Confusion matrix - Train:




## Classification report - Train:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.74 | 0.65 | 0.69 | 307 |
| 1 | 0.86 | 0.91 | 0.89 | 754 |
| accuracy | | | 0.83 | 1061 |
| macro avg | 0.80 | 0.78 | 0.79 | 1061 |
| weighted avg | 0.83 | 0.83 | 0.83 | 1061 |

## Confusion matrix - Test:




## Classification report - Test:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.77 | 0.73 | 0.74 | 153 |
| 1 | 0.86 | 0.89 | 0.88 | 303 |
| accuracy | | | 0.83 | 456 |
| macro avg | 0.82 | 0.81 | 0.81 | 456 |
| weighted avg | 0.83 | 0.83 | 0.83 | 456 |

## Observation:

## Train data:

- Accuracy: 83.41%
- Precision: 86%
- Recall: 91%
- F1-Score: 89%
- AUC: 88.94%

## Test data:

- Accuracy: 83.33%
- Precision: 86%
- Recall: 89%
- F1-Score: 88%
- AUC: 88.76%

## Validness of the model:

- The model is not over-fitted or under-fitted.
- The error in the test data is slightly higher than the train data, which is absolutely fine because the error margin is low and the error in both train and test data is not too high. Thus, the model is not over-fitted or under-fitted

## LDA Model - Tuned:

## Predicted Class and probs:

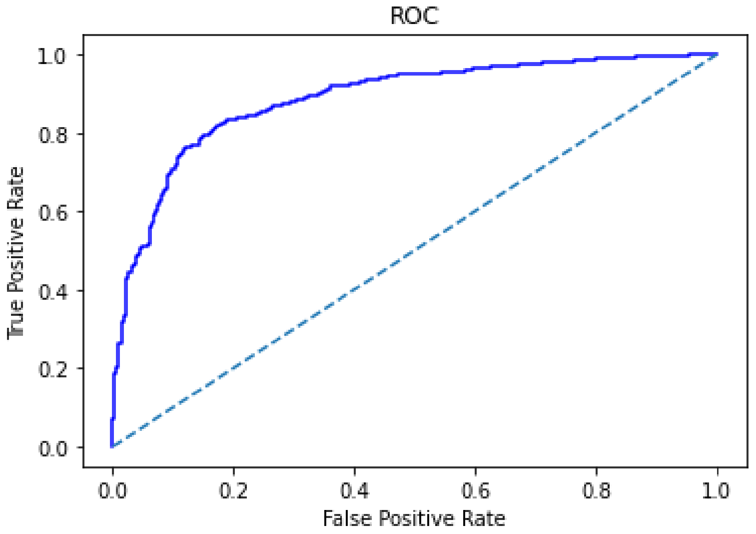
| | 0 | 1 |
|---|---|---|
| 0 | 0.474900 | 0.525100 |
| 1 | 0.146509 | 0.853491 |
| 2 | 0.009523 | 0.990477 |
| 3 | 0.843920 | 0.156080 |
| 4 | 0.069859 | 0.930141 |

## Accuracy - Train:

```
0.8322337417530632
```

## ROC and AUC - Train:

AUC: 0.8868186177520109

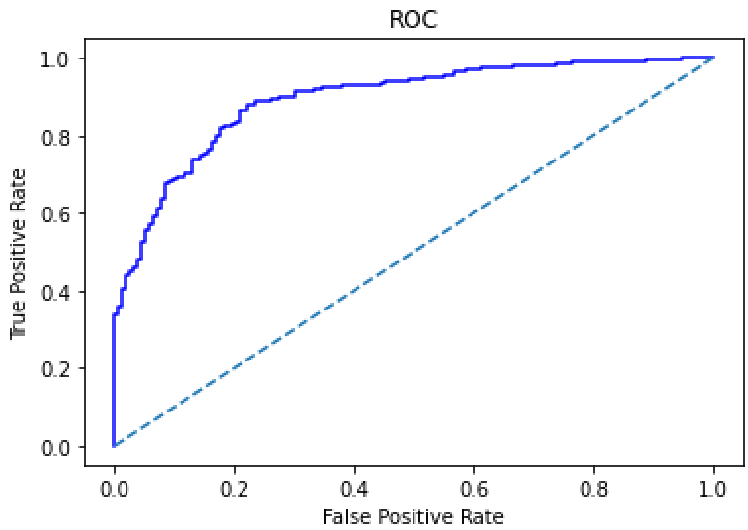



## Accuracy - Test:

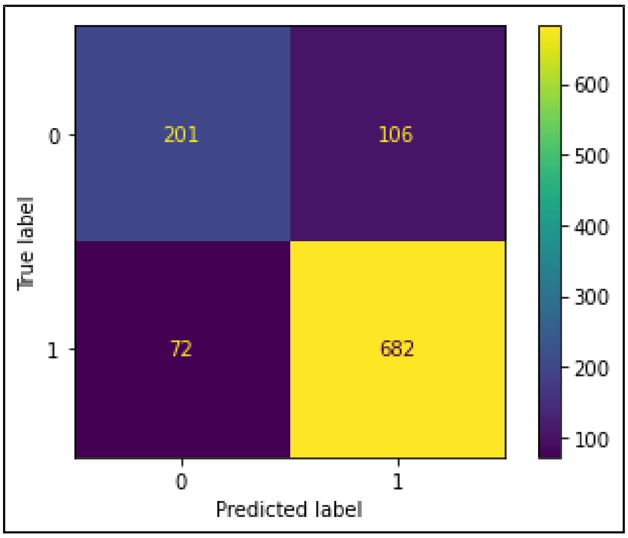
0.8399122807017544

## ROC and AUC - Test:

AUC: 0.8933324705019522



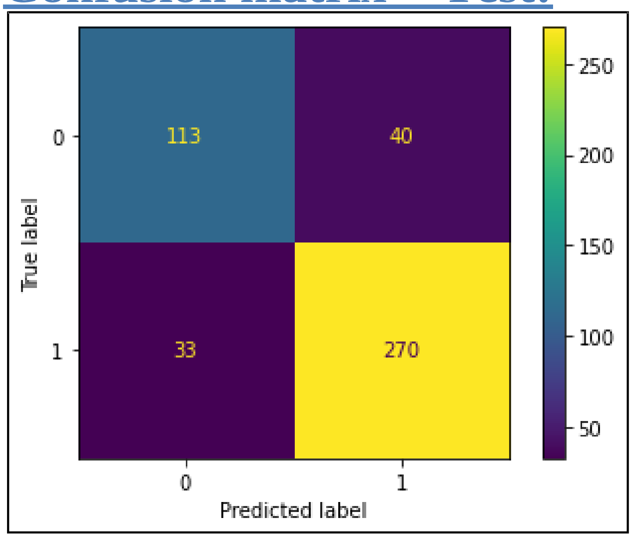
## Confusion matrix - Train:




## Classification report - Train:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.74 | 0.65 | 0.69 | 307 |
| 1 | 0.87 | 0.90 | 0.88 | 754 |
| accuracy | | | 0.83 | 1061 |
| macro avg | 0.80 | 0.78 | 0.79 | 1061 |
| weighted avg | 0.83 | 0.83 | 0.83 | 1061 |

## Confusion matrix - Test:




## Classification report - Test:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.77 | 0.74 | 0.76 | 153 |
| 1 | 0.87 | 0.89 | 0.88 | 303 |
| accuracy | | | 0.84 | 456 |
| macro avg | 0.82 | 0.81 | 0.82 | 456 |
| weighted avg | 0.84 | 0.84 | 0.84 | 456 |

## Observation:

## Train data:

- Accuracy: 83.22%
- Precision: 87%
- Recall: 90%
- F1-Score: 88%
- AUC: 88.68%

## Test data:

- Accuracy: 83.99%
- Precision: 87%
- Recall: 89%
- F1-Score: 88%
- AUC: 89.33%

There is no over-fitting or under-fitting in the tuned LDA model. Overall, it is a good model.

## Comparison between the regular LDA model and tuned LDA model

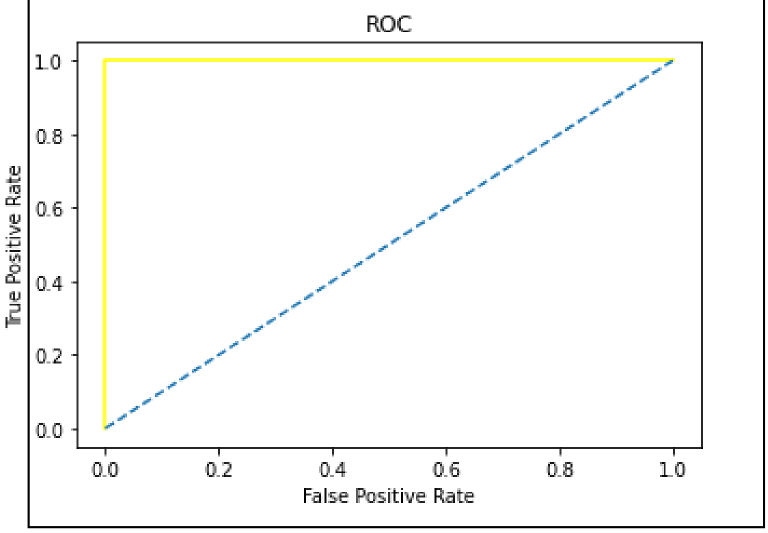
- As we can see, there is not much difference between the performance of regular LDA model and tuned LDA model.
- The values are high overall and there is no over-fitting or under-fitting.
- Therefore both models are equally good models.

## KNN Model - Regular: Predicted Class and probs:

| | 0 | 1 |
|---|---|---|
| 0 | 0.814884 | 0.185116 |
| 1 | 0.123052 | 0.876948 |
| 2 | 0.000000 | 1.000000 |
| 3 | 0.860625 | 0.139375 |
| 4 | 0.126070 | 0.873930 |

ROC and AUC - Train:

```
AUC: 1.0
```

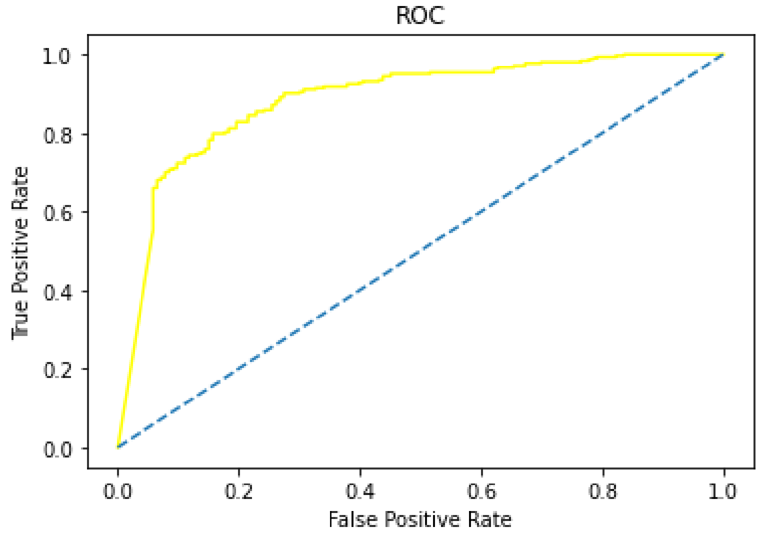



## Accuracy - Test:

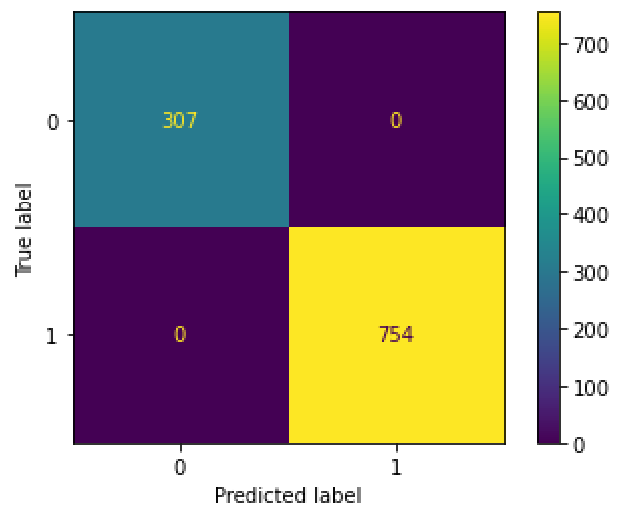
```
0.8377192982456141
```

## ROC and AUC - Test:

AUC: 0.8846178735520611



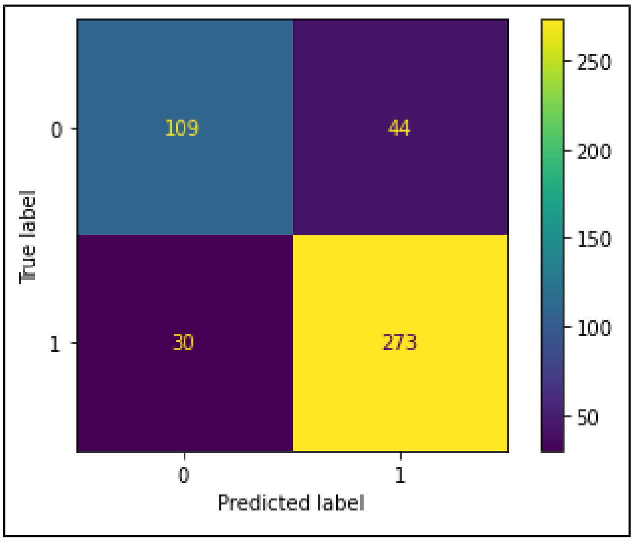

## Confusion matrix - Train:




## Classification report - Train:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 307 |
| 1 | 1.00 | 1.00 | 1.00 | 754 |
| accuracy | | | 1.00 | 1061 |
| macro avg | 1.00 | 1.00 | 1.00 | 1061 |
| weighted avg | 1.00 | 1.00 | 1.00 | 1061 |

## Confusion matrix - Test:




## Classification report - Test:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.78 | 0.71 | 0.75 | 153 |
| 1 | 0.86 | 0.90 | 0.88 | 303 |
| accuracy | | | 0.84 | 456 |
| macro avg | 0.82 | 0.81 | 0.81 | 456 |
| weighted avg | 0.84 | 0.84 | 0.84 | 456 |

## Observation:

## Train data:

- Accuracy: 100%
- Precision: 100%
- Recall: 100%
- F1-Score: 100%
- AUC: 100%

## Test data:

- Accuracy: 83.77%
- Precision: 86%
- Recall: 90%
- F1-Score: 88%
- AUC: 88.46%

## Validness of the model

- The model is over-fitted.
- As we can see, the train data has 100% accuracy and test data has 84% accuracy. The difference is more than 10%. So, we can infer that the KNN model is over-fitted.

## KNN Model - Tuned:

## Predicted Class and probs:

| | 0 | 1 |
|---|---|---|
| 0 | 0.666667 | 0.333333 |
| 1 | 0.142857 | 0.857143 |
| 2 | 0.000000 | 1.000000 |
| 3 | 0.857143 | 0.142857 |
| 4 | 0.190476 | 0.809524 |

## Accuracy - Train:

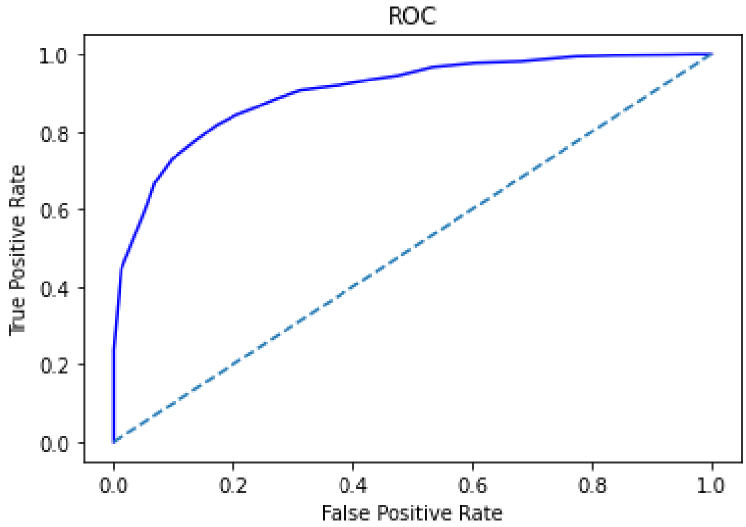
```
0.8435438265786993
```

## Accuracy - Test:

```
0.8618421052631579
```

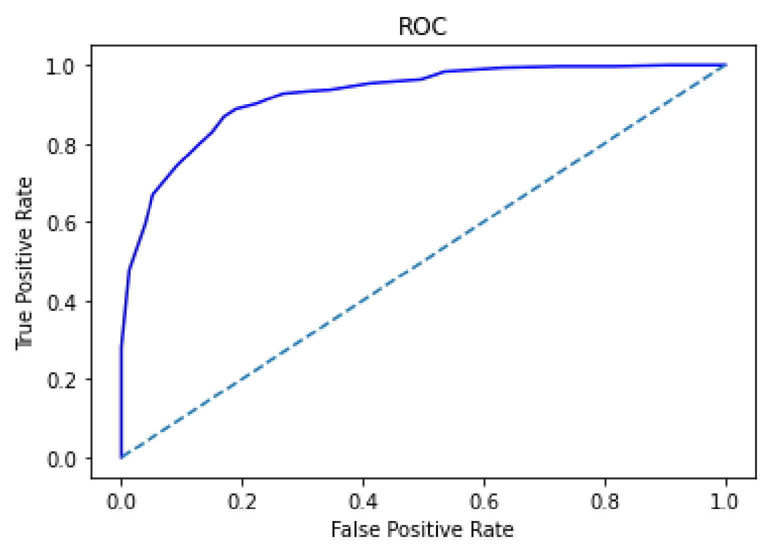
## ROC and AUC - Train:

```
AUC: 0.9022498898383432
```

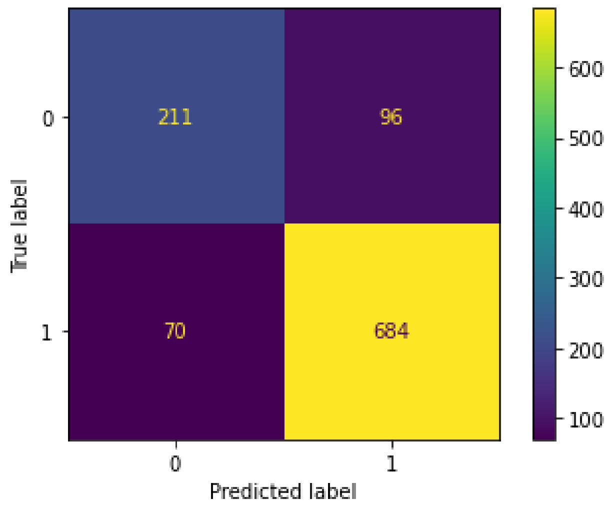


## ROC and AUC - Test:

AUC: 0.922722664423305



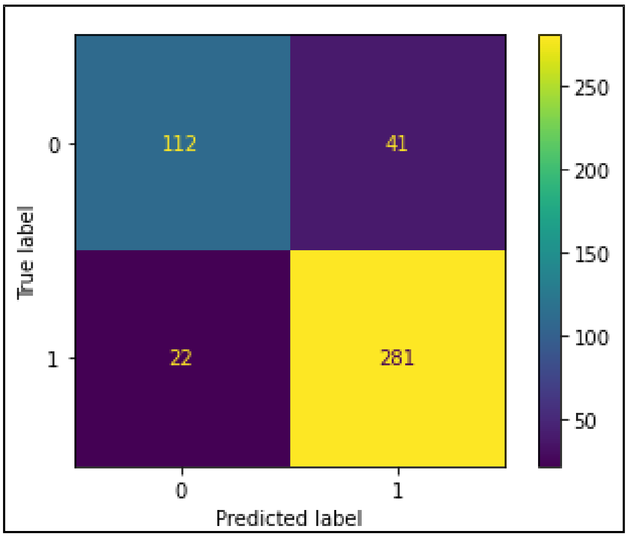

## Confusion matrix - Train:




## Classification report - Train:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.75 | 0.69 | 0.72 | 307 |
| 1 | 0.88 | 0.91 | 0.89 | 754 |
| accuracy | | | 0.84 | 1061 |
| macro avg | 0.81 | 0.80 | 0.80 | 1061 |
| weighted avg | 0.84 | 0.84 | 0.84 | 1061 |

## Confusion matrix - Test:




## Classification report - Test:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.84 | 0.73 | 0.78 | 153 |
| 1 | 0.87 | 0.93 | 0.90 | 303 |
| accuracy | | | 0.86 | 456 |
| macro avg | 0.85 | 0.83 | 0.84 | 456 |
| weighted avg | 0.86 | 0.86 | 0.86 | 456 |

## Observation:

## Train data:

- Accuracy: 84.35%
- Precision: 88%
- Recall: 91%
- F1-Score: 89%
- AUC: 90.23%

## Test data:

- Accuracy: 86.18%
- Precision: 87%
- Recall: 93%
- F1-Score: 90%
- AUC: 92.27%

There is no over-fitting or under-fitting in the tuned KNN model.

Overall, it is a good model.

- Comparison between the regular KNN model and tuned KNN model:
- As we can see, the regular KNN model was over-fitted. But model tuning has helped the model to recover from over-fitting.
- The values are better in the tuned KNN model.
- Therefore, the tuned KNN model is a better model.

## Naïve Bayes Model - Regular:

## Predicted Class and probs:

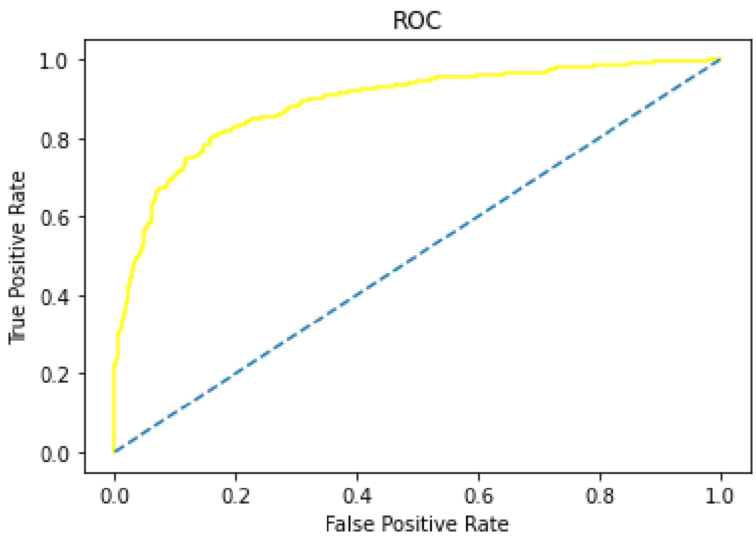
| | 0 | 1 |
|---|---|---|
| 0 | 0.536792 | 0.463208 |
| 1 | 0.120285 | 0.879715 |
| 2 | 0.000332 | 0.999668 |
| 3 | 0.945240 | 0.054760 |
| 4 | 0.039267 | 0.960733 |

## Accuracy - Train:

```
0.8350612629594723
```

## ROC and AUC - Train:

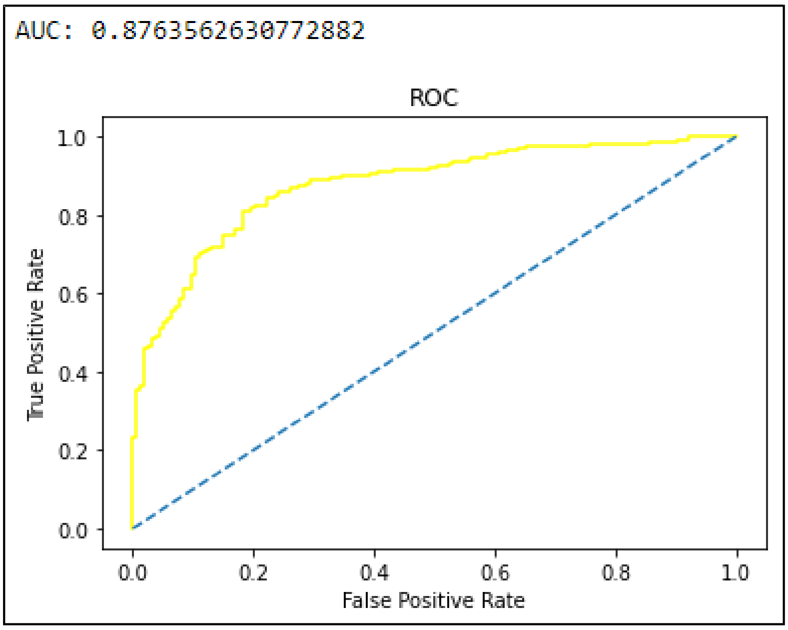
```
AUC: 0.8879375145802193
```



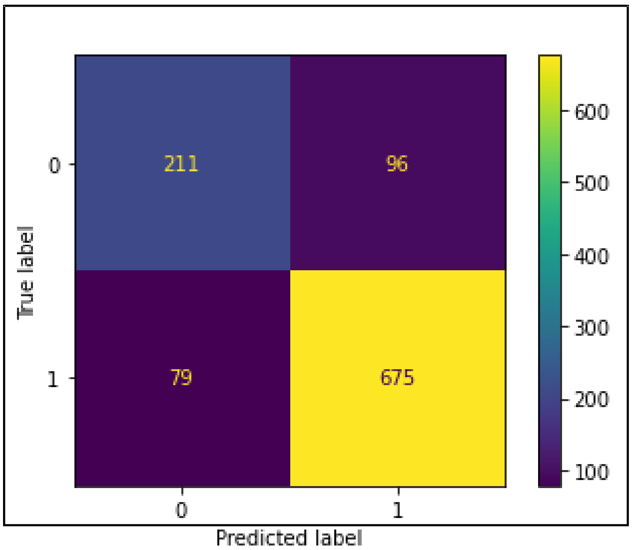

## Accuracy - Test:

```
0.8223684210526315
```

## ROC and AUC - Test:



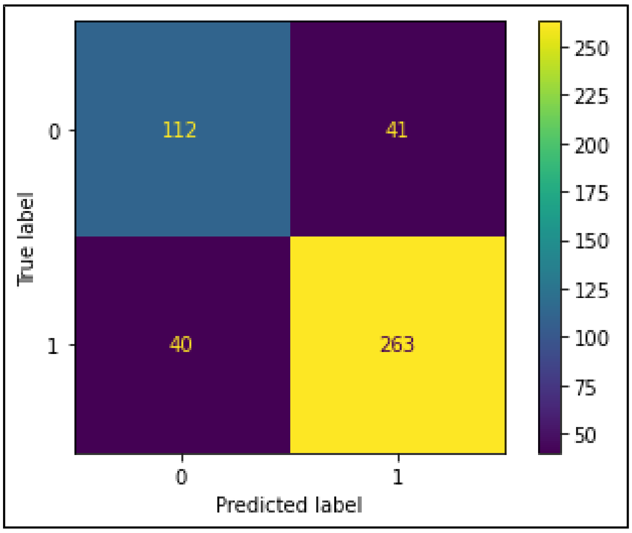

## Confusion matrix - Train:



## Classification report - Train:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.73 | 0.69 | 0.71 | 307 |
| 1 | 0.88 | 0.90 | 0.89 | 754 |
| accuracy | | | 0.84 | 1061 |
| macro avg | 0.80 | 0.79 | 0.80 | 1061 |
| weighted avg | 0.83 | 0.84 | 0.83 | 1061 |

## Confusion matrix - Test:




## Classification report - Test:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.74 | 0.73 | 0.73 | 153 |
| 1 | 0.87 | 0.87 | 0.87 | 303 |
| accuracy | | | 0.82 | 456 |
| macro avg | 0.80 | 0.80 | 0.80 | 456 |
| weighted avg | 0.82 | 0.82 | 0.82 | 456 |

## Observation:

## Train data:

- Accuracy: 83.51%
- Precision: 88%
- Recall: 90%
- F1-Score: 89%AUC: 88.79%

## Test data:

- Accuracy: 82.24%
- Precision: 87%
- Recall: 87%
- F1-Score: 87%
- AUC: 87.64%

## Validness of the model

- The model is not over-fitted or under-fitted.
- The error in the test data is slightly higher than the train data, which is absolutely fine because the error margin is low and the error in both train and test data is not too high. Thus, the model is not over-fitted or under-fitted.
- There is no hyper-parameters to tune in Naive Bayes model. So, we cannot tune this model.

## Ensemble Random Forest Classifier - Regular:

## Predicted Class and probs:

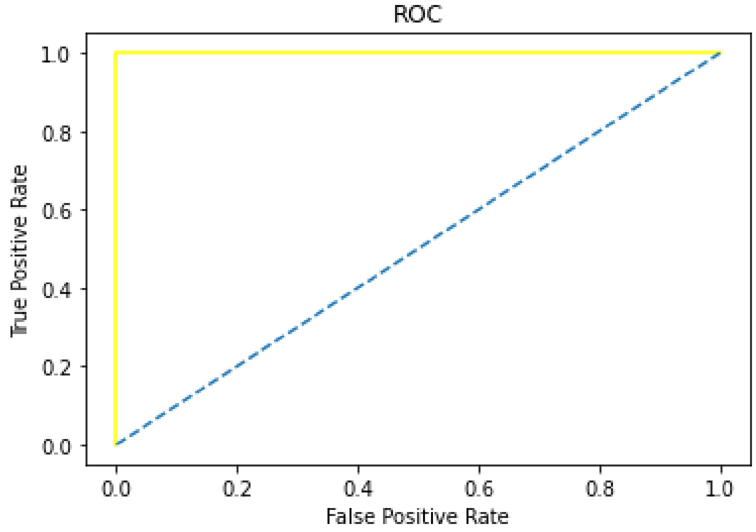
| | 0 | 1 |
|---|---|---|
| 0 | 0.70 | 0.30 |
| 1 | 0.35 | 0.65 |
| 2 | 0.00 | 1.00 |
| 3 | 0.80 | 0.20 |
| 4 | 0.14 | 0.86 |

## Accuracy - Train:

```
1.0
```

## ROC and AUC - Train:

```
AUC: 1.0
```

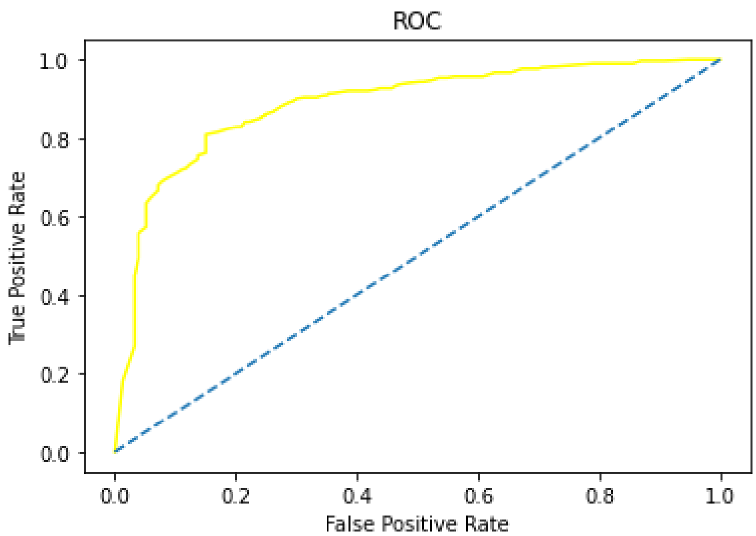



## Accuracy - Test:

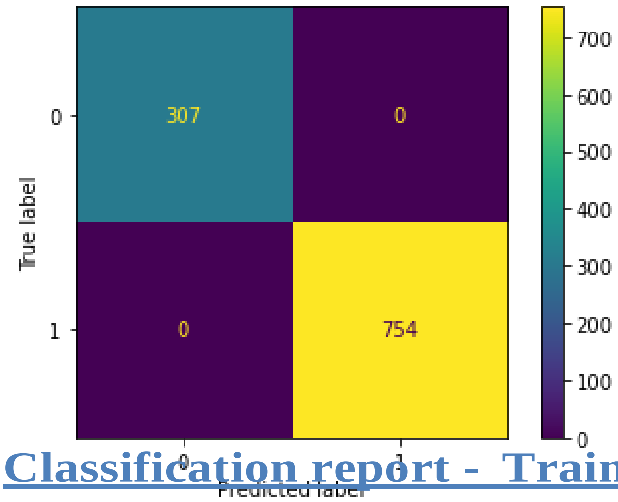
```
0.8289473684210527
```

## ROC and AUC - Test:

```
AUC: 0.8862033262149743
```



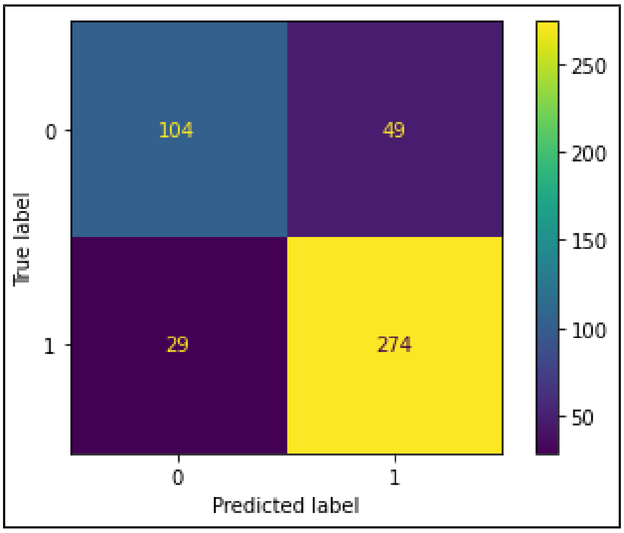

## Confusion matrix - Train:




## Classification report - Train:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 1.00 | 1.00 | 307 |
| 1 | 1.00 | 1.00 | 1.00 | 754 |
| accuracy | | | 1.00 | 1061 |
| macro avg | 1.00 | 1.00 | 1.00 | 1061 |
| weighted avg | 1.00 | 1.00 | 1.00 | 1061 |

## Confusion matrix - Test:



## Classification report - Test:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.78 | 0.68 | 0.73 | 153 |
| 1 | 0.85 | 0.90 | 0.88 | 303 |
| accuracy | | | 0.83 | 456 |
| macro avg | 0.82 | 0.79 | 0.80 | 456 |
| weighted avg | 0.83 | 0.83 | 0.83 | 456 |

## Observation:

## Train data:

- Accuracy: 100%
- Precision: 100%
- Recall: 100%
- F1-Score: 100%
- AUC: 100%

## Test data:

- Accuracy: 82.68%
- Precision: 84%
- Recall: 91%
- F1-Score: 88%
- AUC: 88.62%

The model is over-fitted. So we will use bagging to improve the performance of the model.

## Bagging - Regular:

## Predicted Class and probs:

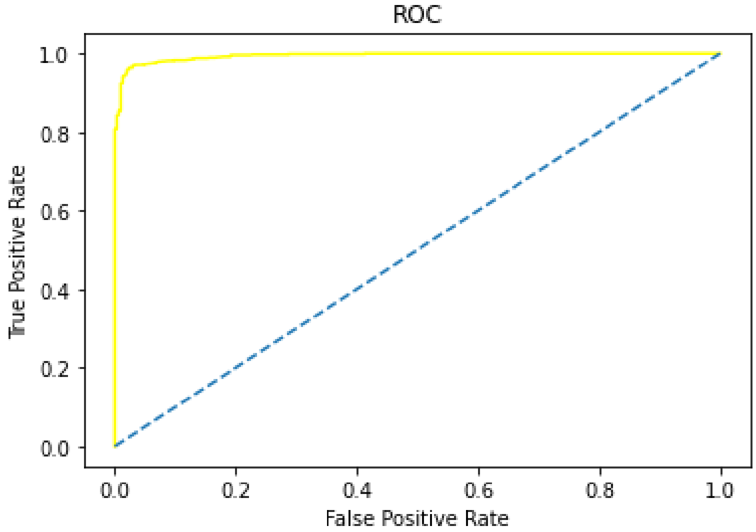
| | 0 | 1 |
|---|---|---|
| 0 | 0.668 | 0.332 |
| 1 | 0.280 | 0.720 |
| 2 | 0.054 | 0.946 |
| 3 | 0.812 | 0.188 |
| 4 | 0.130 | 0.870 |

## Accuracy - Train:

```
0.9538171536286523
```

## ROC and AUC - Train:

```
AUC: 0.9940771909209515
```

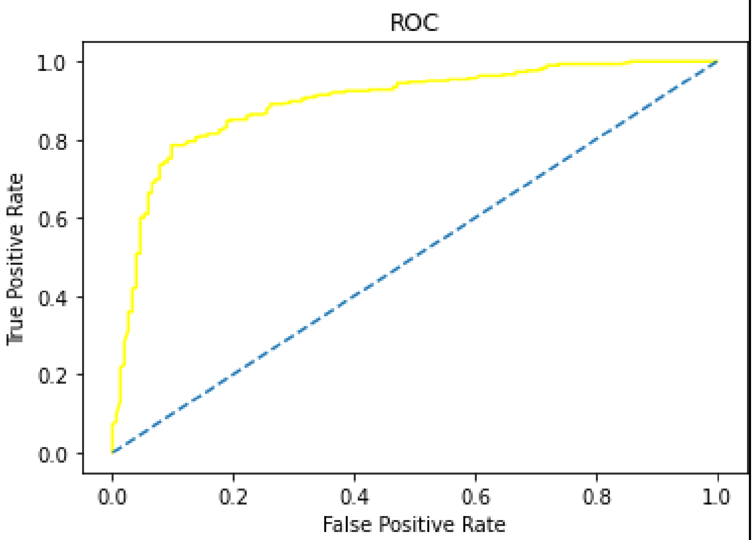



## Accuracy - Test:

```
0.8245614035087719
```

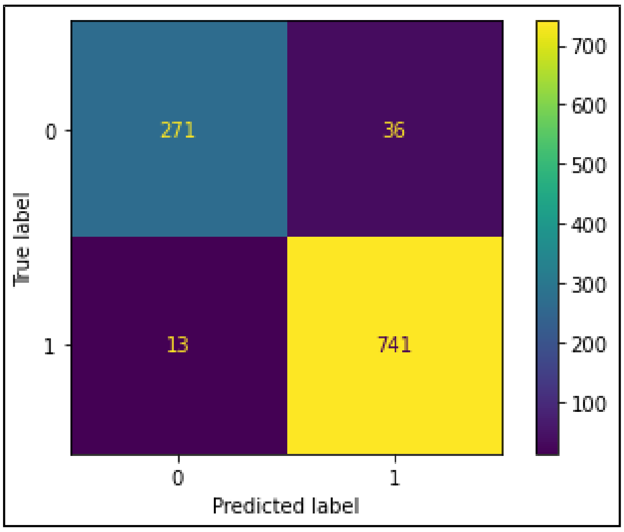
## ROC and AUC –

```
AUC: 0.8940011648223646
```



## Test:

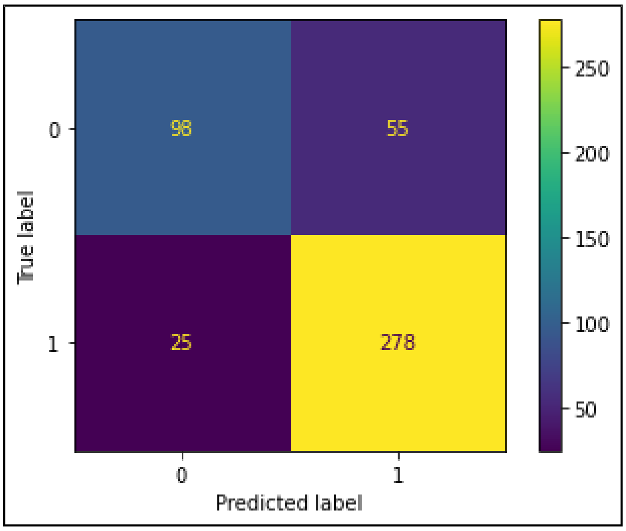
## Confusion matrix - Train:




## Classification report - Train:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.95 | 0.88 | 0.92 | 307 |
| 1 | 0.95 | 0.98 | 0.97 | 754 |
| accuracy | | | 0.95 | 1061 |
| macro avg | 0.95 | 0.93 | 0.94 | 1061 |
| weighted avg | 0.95 | 0.95 | 0.95 | 1061 |

## Confusion matrix - Test:



## Classification report - Test:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.80 | 0.64 | 0.71 | 153 |
| 1 | 0.83 | 0.92 | 0.87 | 303 |
| accuracy | | | 0.82 | 456 |
| macro avg | 0.82 | 0.78 | 0.79 | 456 |
| weighted avg | 0.82 | 0.82 | 0.82 | 456 |

## Observation:

## Train data:

- Accuracy: 95.38%
- Precision: 95%
- Recall: 98%
- F1-Score: 97%
- AUC: 99.4%

## Test data:

- Accuracy: 82.46%
- Precision: 83%
- Recall: 92%
- F1-Score: 87%
- AUC: 89.4%

After using bagging, the model is still over-fitted. The values are high. But the difference between the train and test accuracy is high.

## Bagging - Tuned: Predicted

## Class and probs:

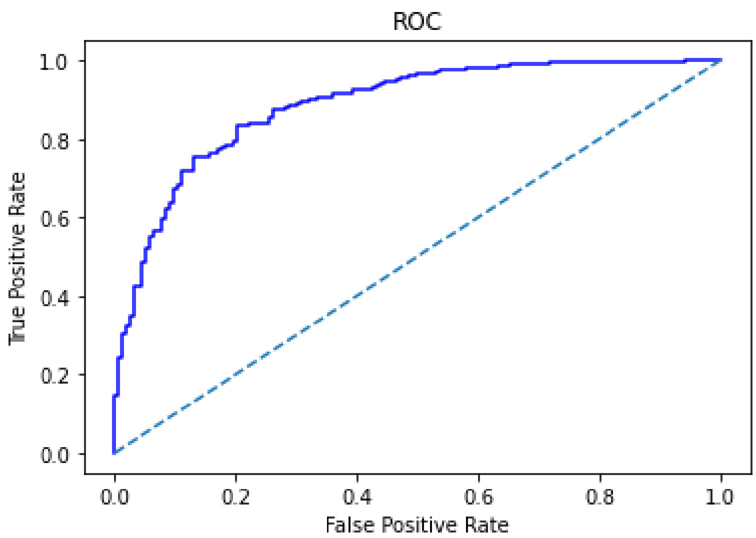
| | 0 | 1 |
|---|---|---|
| 0 | 0.422545 | 0.577455 |
| 1 | 0.193451 | 0.806549 |
| 2 | 0.025707 | 0.974293 |
| 3 | 0.782613 | 0.217387 |
| 4 | 0.312266 | 0.687734 |

## Accuracy - Test:

```
0.8135964912280702
```

## ROC and AUC - Test:

```
AUC: 0.8858042667011798
```

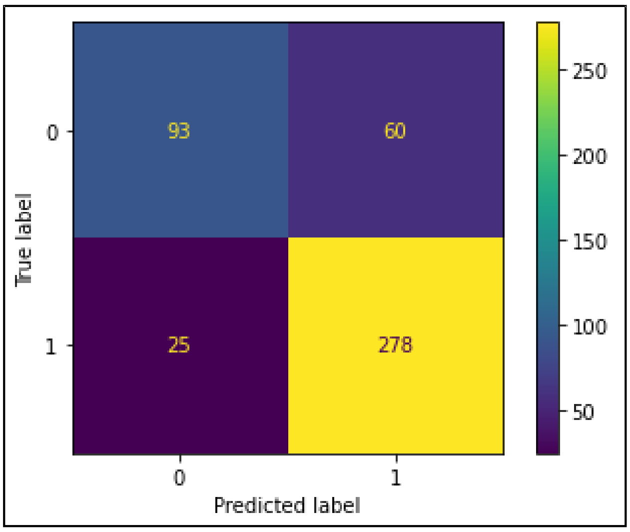



## Confusion matrix - Train:

## Classification report - Train:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.81 | 0.61 | 0.69 | 307 |
| 1 | 0.86 | 0.94 | 0.90 | 754 |
| accuracy | | | 0.84 | 1061 |
| macro avg | 0.83 | 0.77 | 0.79 | 1061 |
| weighted avg | 0.84 | 0.84 | 0.84 | 1061 |

## Confusion matrix - Test:



## Classification report - Test:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.79 | 0.61 | 0.69 | 153 |
| 1 | 0.82 | 0.92 | 0.87 | 303 |
| accuracy | | | 0.81 | 456 |
| macro avg | 0.81 | 0.76 | 0.78 | 456 |
| weighted avg | 0.81 | 0.81 | 0.81 | 456 |

## Observation:

## Train data:

- Accuracy: 84.45%
- Precision: 86%
- Recall: 94%
- F1-Score: 90%
- AUC: 90.41%

## Test data:

- Accuracy: 81.36%
- Precision: 82%
- Recall: 92%
- F1-Score: 87%
- AUC: 88.58%

The tuning of the model has help the model recover from over-fitting. Now the model is a good model.

## AdaBoosting - Regular:

## Predicted Class and probs:

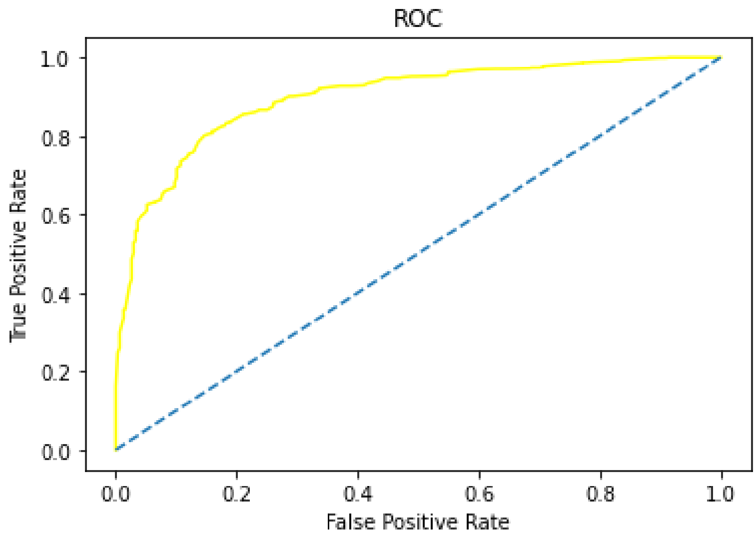
| | 0 | 1 |
|---|---|---|
| 0 | 0.514483 | 0.485517 |
| 1 | 0.452083 | 0.547917 |
| 2 | 0.312219 | 0.687781 |
| 3 | 0.564895 | 0.435105 |
| 4 | 0.459939 | 0.540061 |

## Accuracy - Train:

```
0.8426013195098964
```

## ROC and AUC - Train:

```
AUC: 0.897912544604671
```

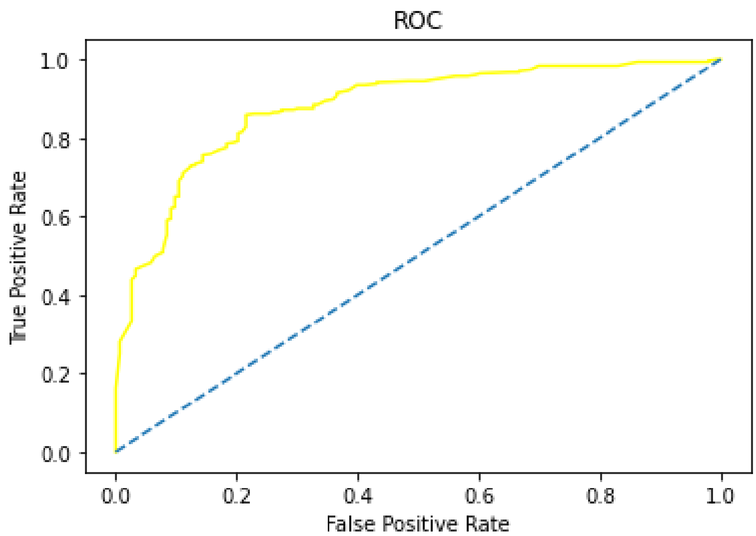



## Accuracy - Test:

```
0.8201754385964912
```

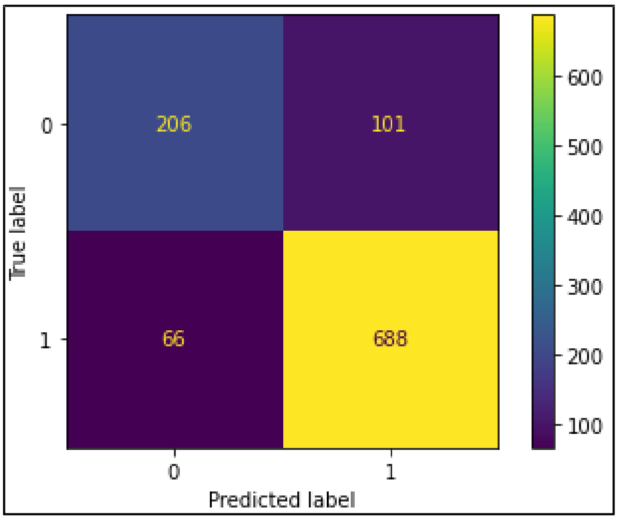
## ROC and AUC – Test

```
AUC: 0.8781358528009664
```




:

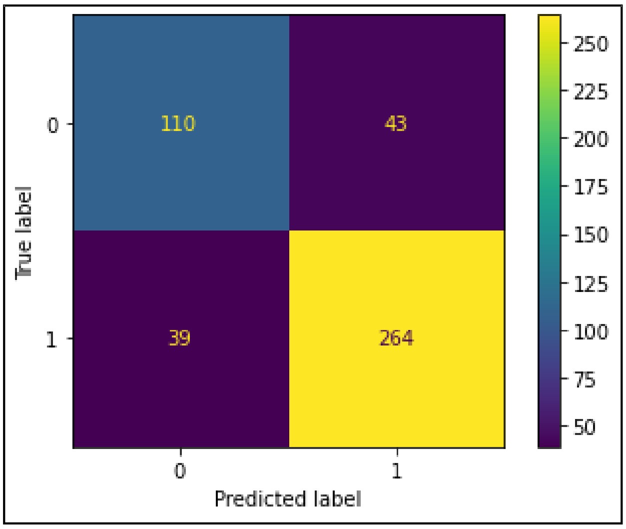
## Confusion matrix - Train:




## Classification report - Train:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.76 | 0.67 | 0.71 | 307 |
| 1 | 0.87 | 0.91 | 0.89 | 754 |
| accuracy | | | 0.84 | 1061 |
| macro avg | 0.81 | 0.79 | 0.80 | 1061 |
| weighted avg | 0.84 | 0.84 | 0.84 | 1061 |

## Confusion matrix - Test:



## Classification report - Test:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.74 | 0.72 | 0.73 | 153 |
| 1 | 0.86 | 0.87 | 0.87 | 303 |
| accuracy | | | 0.82 | 456 |
| macro avg | 0.80 | 0.80 | 0.80 | 456 |
| weighted avg | 0.82 | 0.82 | 0.82 | 456 |

## Observation:

## Train data:

- Accuracy: 84.26%
- Precision: 87%
- Recall: 91%
- F1-Score: 89%
- AUC: 89.79%

## Test data:

- Accuracy: 82.02%
- Precision: 86%
- Recall: 87%
- F1-Score: 87%
- AUC: 87.81%

The tuning of the model has help the model recover from over-fitting. Now the model is a good model.

## AdaBoosting - Tuned:

## Predicted Class and probs:

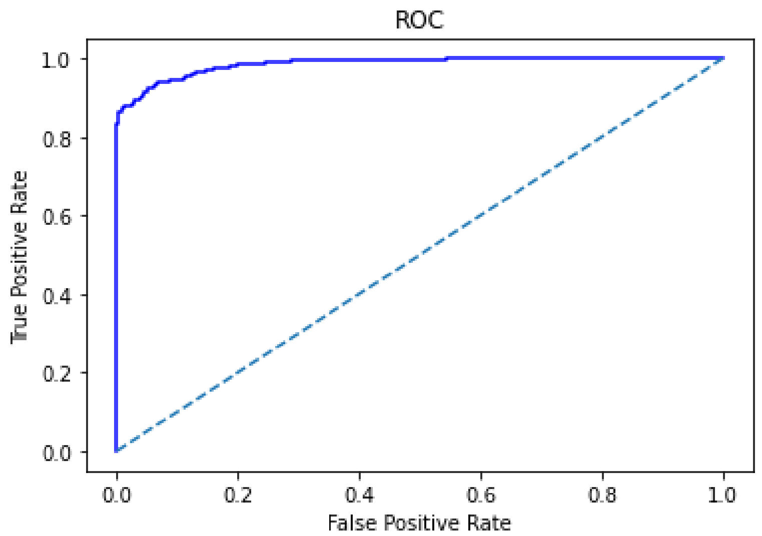
| | 0 | 1 |
|---|---|---|
| 0 | 0.545025 | 0.454975 |
| 1 | 0.505622 | 0.494378 |
| 2 | 0.203440 | 0.796560 |
| 3 | 0.553043 | 0.446957 |
| 4 | 0.378112 | 0.621888 |

## Accuracy - Train:

```
0.9349670122525919
```

## ROC and AUC - Train:

```
AUC: 0.9862276328635983
```

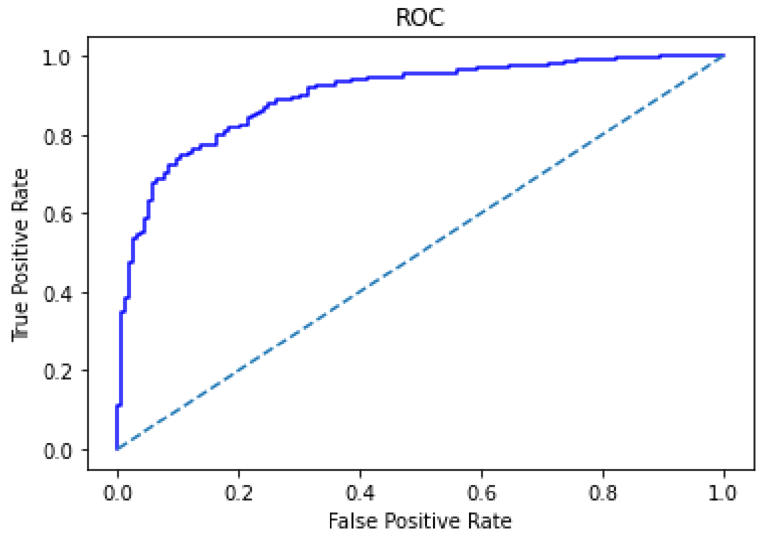



## Accuracy - Test:

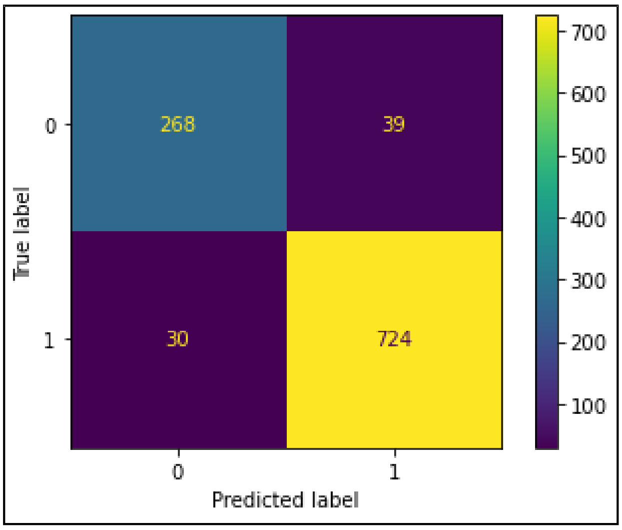
```
0.831140350877193
```

## ROC and AUC - Test:

```
AUC: 0.8998684182143705
```



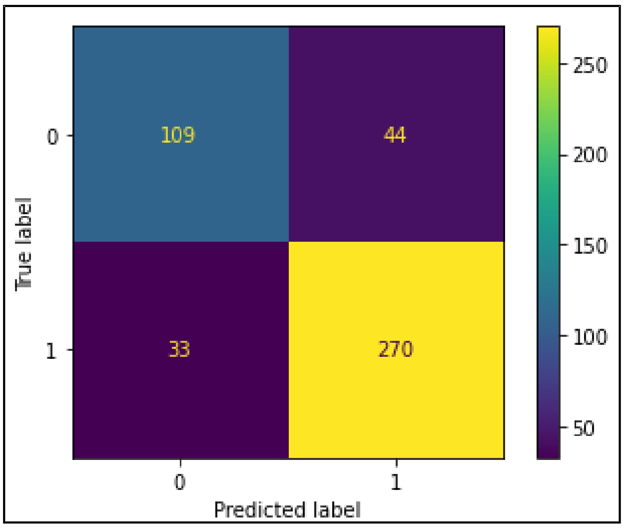
## Confusion matrix - Train:




## Classification report - Train:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 0.87 | 0.89 | 307 |
| 1 | 0.95 | 0.96 | 0.95 | 754 |
| accuracy | | | 0.93 | 1061 |
| macro avg | 0.92 | 0.92 | 0.92 | 1061 |
| weighted avg | 0.93 | 0.93 | 0.93 | 1061 |

## Confusion matrix - Test:



## Classification report - Test:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.77 | 0.71 | 0.74 | 153 |
| 1 | 0.86 | 0.89 | 0.88 | 303 |
| accuracy | | | 0.83 | 456 |
| macro avg | 0.81 | 0.80 | 0.81 | 456 |
| weighted avg | 0.83 | 0.83 | 0.83 | 456 |

## Observation:

## Train data:

- Accuracy: 93.5%
- Precision: 95%
- Recall: 96%
- F1-Score: 95%
- AUC: 98.62%

## Test data:

- Accuracy: 83.11%
- Precision: 86%
- Recall: 89%
- F1-Score: 88%
- AUC: 89.99%

The model is a good model. There is no over-fitting. There is improvement from the regular model.

## Gradient Boosting - Regular:

## Predicted Class and probs:

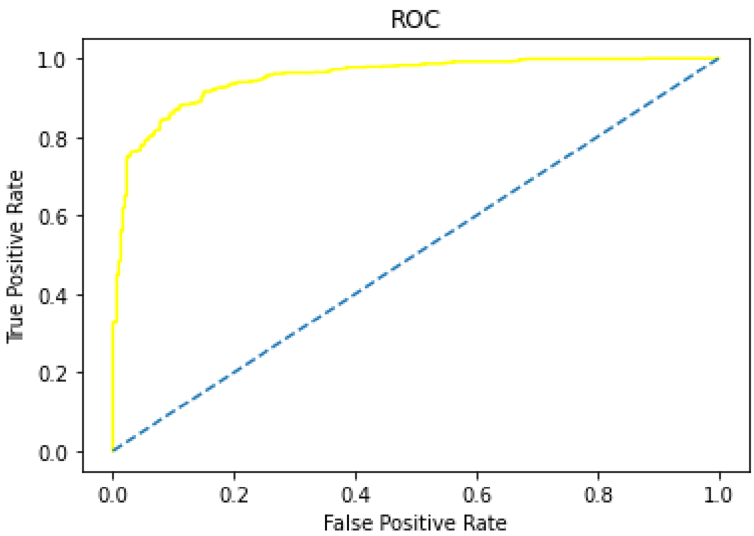
| | 0 | 1 |
|---|---|---|
| 0 | 0.690657 | 0.309343 |
| 1 | 0.236942 | 0.763058 |
| 2 | 0.001102 | 0.998898 |
| 3 | 0.840247 | 0.159753 |
| 4 | 0.111644 | 0.888356 |

## Accuracy - Train:

```
0.8925541941564562
```

## ROC and AUC - Train:

```
AUC: 0.951155185373988
```

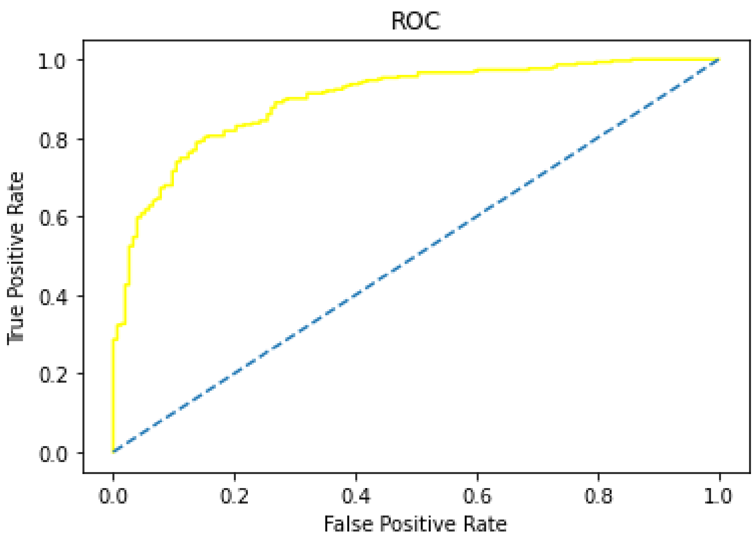



## Accuracy - Test:

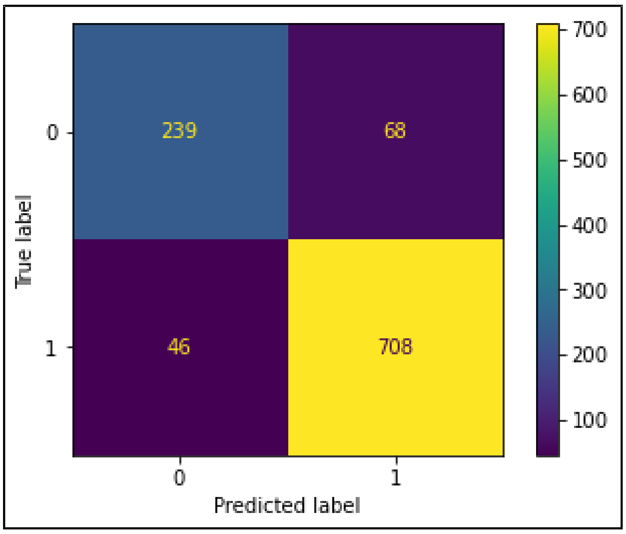
```
0.8333333333333334
```

## ROC and AUC - Test:

```
AUC: 0.8987143812420458
```



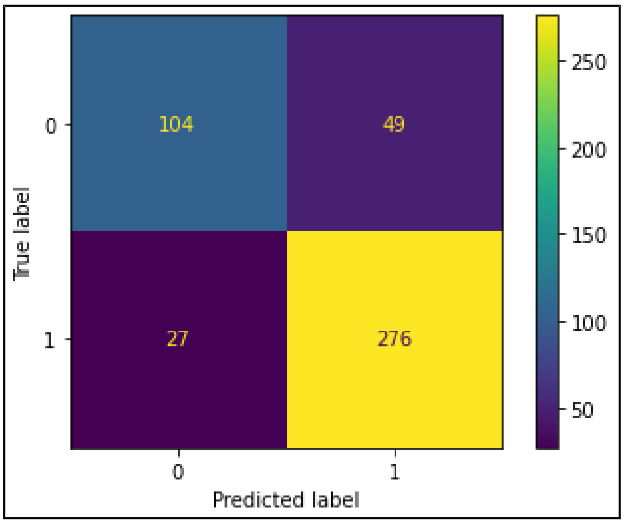
## Confusion matrix - Train:




## Classification report - Train:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.84 | 0.78 | 0.81 | 307 |
| 1 | 0.91 | 0.94 | 0.93 | 754 |
| accuracy | | | 0.89 | 1061 |
| macro avg | 0.88 | 0.86 | 0.87 | 1061 |
| weighted avg | 0.89 | 0.89 | 0.89 | 1061 |

## Confusion matrix - Test:



## Classification report - Test:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.79 | 0.68 | 0.73 | 153 |
| 1 | 0.85 | 0.91 | 0.88 | 303 |
| accuracy | | | 0.83 | 456 |
| macro avg | 0.82 | 0.80 | 0.81 | 456 |
| weighted avg | 0.83 | 0.83 | 0.83 | 456 |

## Observation:

## Train data:

- Accuracy: 89.26%
- Precision: 91%
- Recall: 94%
- F1-Score: 93%
- AUC: 95.11%

## Test data:

- Accuracy: 83.33%
- Precision: 85%
- Recall: 91%
- F1-Score: 88%
- AUC: 89.87%

The values are high. There is no over-fitting of any sorts. Themodel is a good model.

## Gradient Boosting - Tuned:

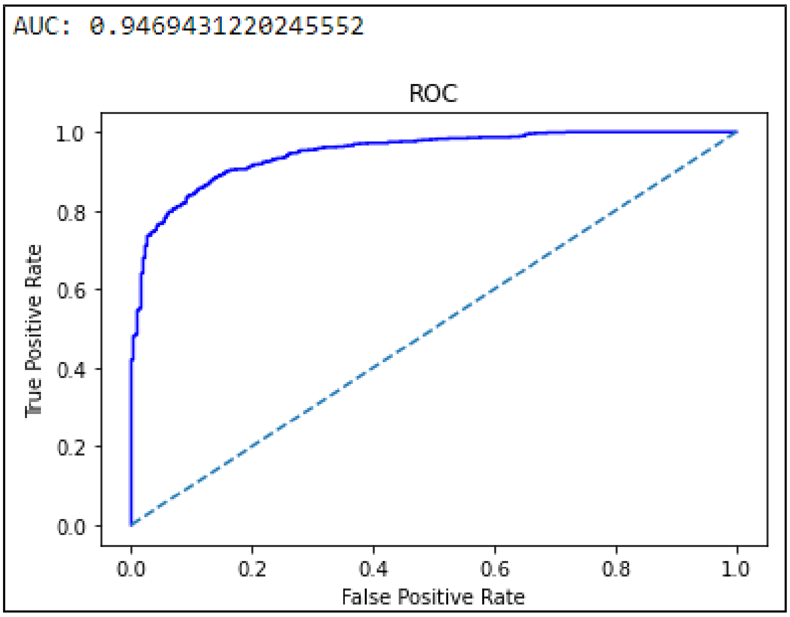
## Predicted Class and probs:

| | 0 | 1 |
|---|---|---|
| 0 | 0.554136 | 0.445864 |
| 1 | 0.314606 | 0.685394 |
| 2 | 0.040688 | 0.959312 |
| 3 | 0.779993 | 0.220007 |
| 4 | 0.165782 | 0.834218 |

## Accuracy - Train:

```
0.883129123468426
```

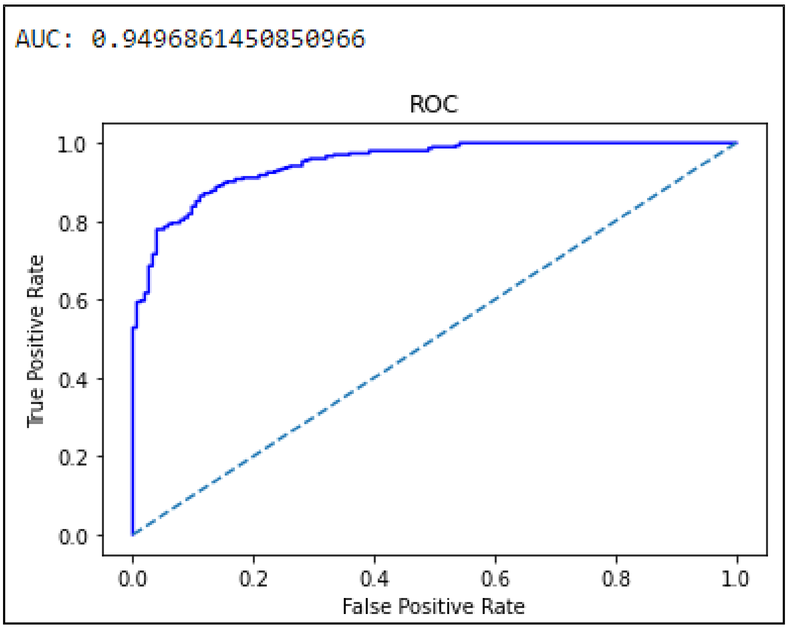
## ROC and AUC - Train:



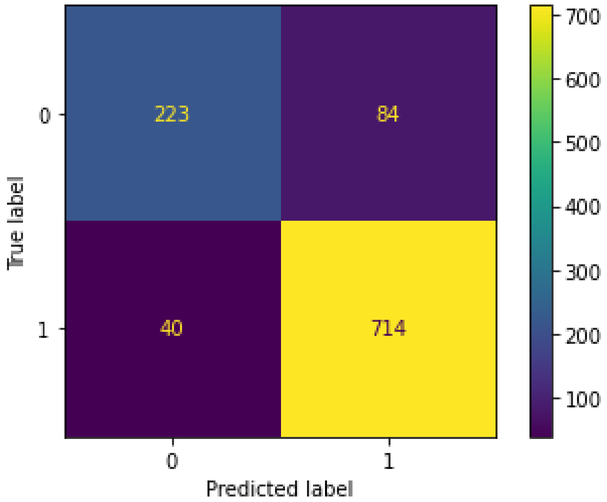

## Accuracy - Test:

```
0.8728070175438597
```

## ROC and AUC - Test:



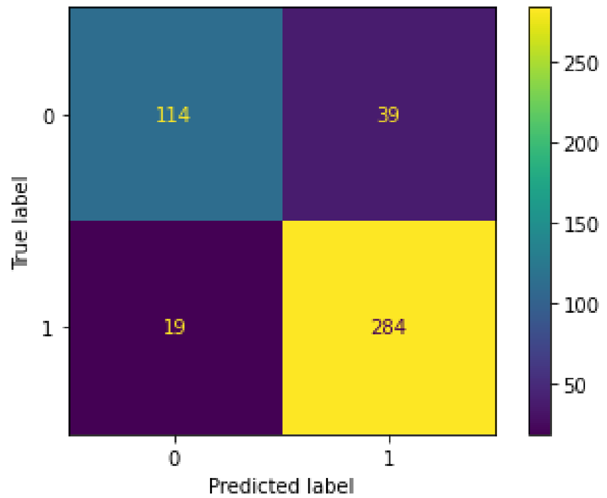
## Confusion matrix - Train:




## Classification report - Train:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.85 | 0.73 | 0.78 | 307 |
| 1 | 0.89 | 0.95 | 0.92 | 754 |
| accuracy | | | 0.88 | 1061 |
| macro avg | 0.87 | 0.84 | 0.85 | 1061 |
| weighted avg | 0.88 | 0.88 | 0.88 | 1061 |

## Confusion matrix - Test:



## Classification report - Test:

```
              precision    recall  f1-score   support

           0       0.86      0.75      0.80       153
           1       0.88      0.94      0.91       303

    accuracy                           0.87       456
   macro avg       0.87      0.84      0.85       456
weighted avg       0.87      0.87      0.87       456
```

## Observation:

## Train data:

- Accuracy: 88.31%
- Precision: 89%
- Recall: 95%
- F1-Score: 92%
- AUC: 94.69%

## Test data:

- Accuracy: 87.28%
- Precision: 88%
- Recall: 94%
- F1-Score: 91%
- AUC: 94.97%

The tuning of the Gradient Boost model has improved the model further. The values are high. The better is better than the regular model.

## Comparison of train data of all models in a structured tabular manner:

| | Accuracy | Precision | Recall | F1-Score | AUC |
|---|---|---|---|---|---|
| LR - Regular | 83.41% | 86% | 92% | 89% | 88.98 % |
| LR - Tuned | 83.6% | 86% | 92% | 89% | 88.89 % |

| | | | | | |
|---|---|---|---|---|---|
| LDA - Regular | 83.41% | 86% | 91% | 89% | 88.94 % |
| LDA - Tuned | 83.22% | 87% | 90% | 88% | 88.68 % |
| KNN - Regular | 100% | 100% | 100% | 100% | 100% |
| KNN - Tuned | 84.35% | 88% | 91% | 89% | 90.23 % |
| Naïve Bayes - Regular | 83.51% | 88% | 90% | 89% | 88.79 % |
| Random Forest - Regular | 100% | 100% | 100% | 100% | 100% |
| Bagging - Regular | 95.38% | 95% | 98% | 97% | 99.4% |
| Bagging - Tuned | 84.45% | 86% | 94% | 90% | 90.41 % |
| AdaBoosting - Regular | 84.26% | 87% | 91% | 89% | 89.79 % |
| AdaBoosting - Tuned | 93.5% | 95% | 96% | 95% | 98.62 % |
| Gradient Boosting - Regular | 89.26% | 91% | 94% | 93% | 95.11 % |
| Gradient Boosting - Tuned | 88.31% | 89% | 95% | 92% | 94.69 % |

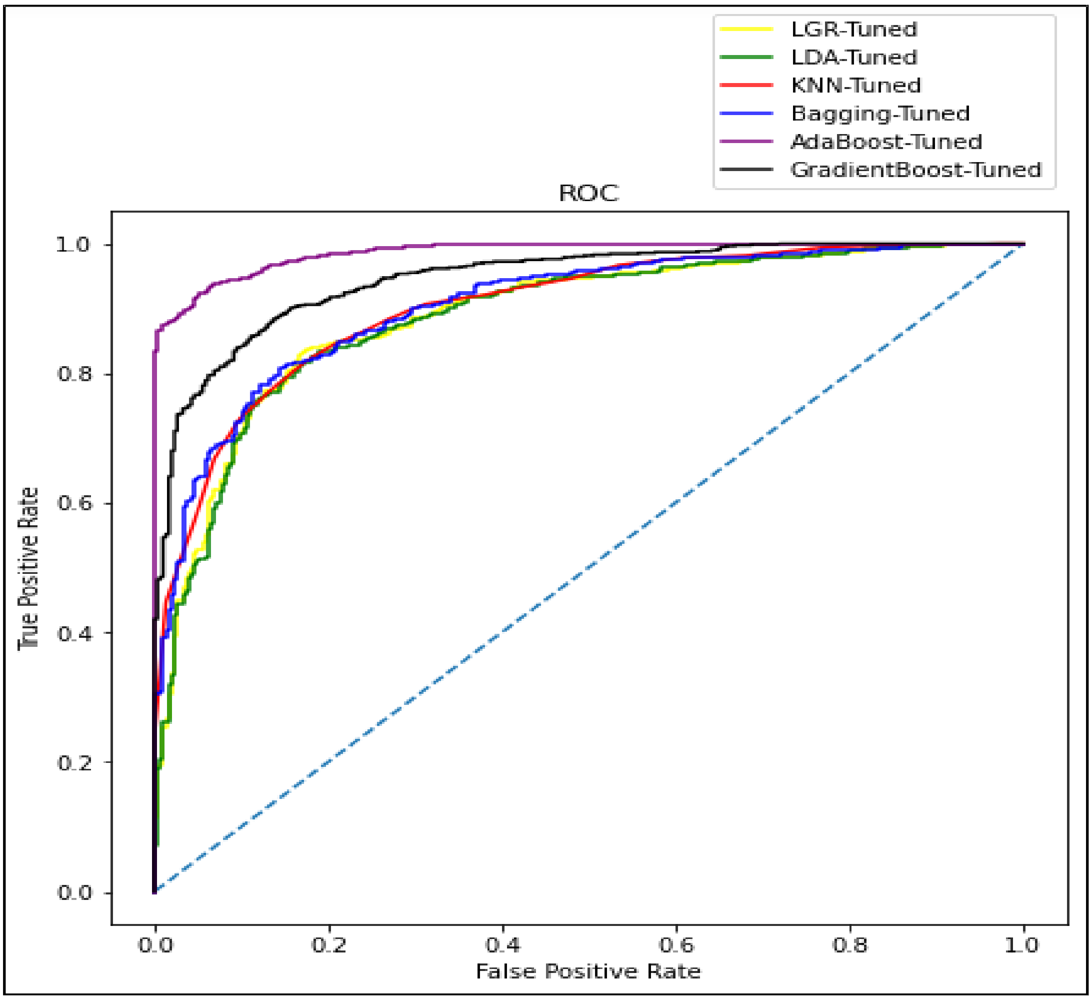
## Comparison of test data of all models in a structured tabular manner:

| | Accuracy | Precision | Recall | F1-Score | AUC |
|---|---|---|---|---|---|
| LR - Regular | 82.68% | 86% | 89% | 87% | 88.4% |
| LR - Tuned | 84.21% | 86% | 90% | 88% | 89.05 % |
| LDA - Regular | 83.33% | 86% | 89% | 88% | 88.76 % |
| LDA - Tuned | 83.99% | 87% | 89% | 88% | 89.33 % |
| KNN - Regular | 83.77% | 86% | 90% | 88% | 88.46 % |
| KNN - Tuned | 86.18% | 87% | 93% | 90% | 92.27 % |
| Naïve Bayes - Regular | 82.24% | 87% | 87% | 87% | 87.64 |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | % |
| Random Forest - Regular | 82.68% | 84% | 91% | 88% | 88.62 % |
| Bagging - Regular | 82.46% | 83% | 92% | 87% | 89.4% |
| Bagging - Tuned | 81.36% | 82% | 92% | 87% | 88.58 % |
| AdaBoosting - Regular | 82.02% | 86% | 87% | 87% | 87.81 % |
| AdaBoosting - Tuned | 83.11% | 86% | 89% | 88% | 89.99 % |
| Gradient Boosting - Regular | 83.33% | 85% | 91% | 88% | 89.87 % |
| Gradient Boosting - Tuned | 87.28% | 88% | 94% | 91% | 94.97 % |

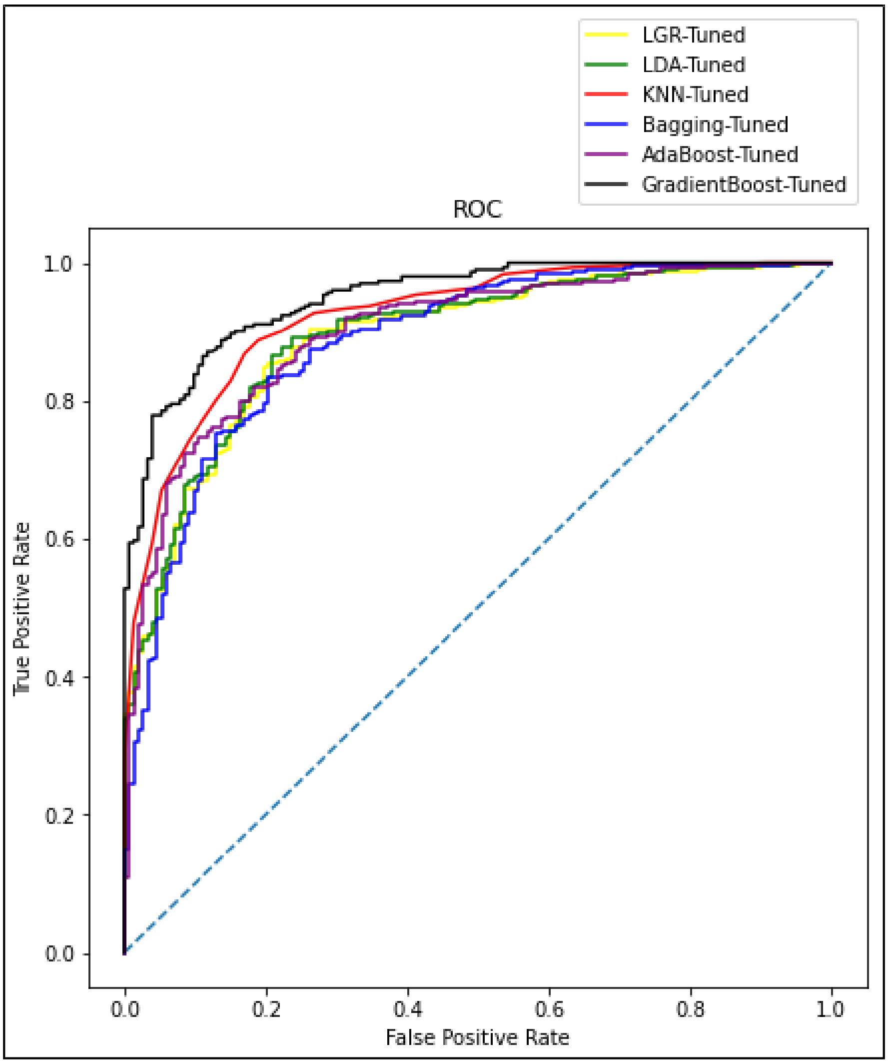
## Comparing the AUC, ROC curve on the train data of all the tuned models:

In all the models, tuned ones are better than the regular models. So, we compare only the tuned models and describe which model is the best/optimized.



## Comparing the AUC, ROC curve on the test data of all the tuned models:

In all the models, tuned ones are better than the regular models. So, we compare only the tuned models and describe which model is the best/optimized.




## Conclusion:

- There is no under-fitting or over-fitting in any of the tuned models.

- All the tuned models have high values and every model is good. But as we can see, the most consistent tuned model in both train and test data is the Gradient Boost model.

- The tuned gradient boost model performs the best with 88.31% accuracy score in train and 87.28% accuracy score in test. Also it has the best AUC score of 94% in both train and test data which is the highest of all the models.
- It also has a precision score of 88% and recall of 94% which is also the highest of all the models. So, we conclude that Gradient Boost Tuned model is the best/optimized model.

**1.8) 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.**

- Labour party has more than double the votes of conservative party.
- Most number of people have given a score of 3 and 4 for the national economic condition and the average score is 3.245221
- Most number of people have given a score of 3 and 4 for the household economic condition and the average score is 3.137772
- Blair has higher number of votes than Hague and the scores are much better for Blair than for Hague.
- The average score of Blair is 3.335531 and the average score of Hague is 2.749506. So, here we can see that,Blair has a better score.
- On a scale of 0 to 3, about 30% of the total population has zero knowledge about politics/parties.
- People who gave a low score of 1 to a certain party, still decided to vote for the same party instead of voting for the other party. This can be because of lack of political knowledge among the people.
- People who have higher Eurosceptic sentiment, has voted for the conservative party and lower the Eurosceptic sentiment, higher the votes for Labour party.
- Out of 454 people who gave a score of 0 for political knowledge, 360 people have voted for the labour party and 94 people have voted for the conservative party.
- All models performed well on training data set as well as test dataset. The tuned models have performed better than the regular models.
- There is no over-fitting in any model except Random Forest and Bagging regular models.

- Gradient Boosting model tuned is the best/optimized model.

## Business recommendations:

- Hyper-parameters tuning is an import aspect of model building. There are limitations to this as to process these combinations, huge amount of processing power is required. But if tuning can be done with many sets of parameters, we might get even better results.
- Gathering more data will also help in training the models and thus improving the predictive powers.
- We can also create a function in which all the models predict the outcome in sequence. This will helps in better understanding and the probability of what the outcome will be.
- Using Gradient Boosting model without scaling for predicting the outcome as it has the best optimized performance.

## Problem 2:

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

President Franklin D. Roosevelt in 1941

President John F. Kennedy in 1961 President

Richard Nixon in 1973

**2.1) Find the number of characters, words and sentences for the mentioned documents. (Hint: use .words(), .raw(), .sent() for extracting counts)**

## Number of characters:

| | Speech | char_count |
|---|---|---|
| 0 | On each national day of inauguration since 178... | 7571 |
| 1 | Vice President Johnson, Mr. Speaker, Mr. Chief... | 7618 |
| 2 | Mr. Vice President, Mr. Speaker, Mr. Chief Jus... | 9991 |

- President Franklin D. Roosevelt's speech have 7571 characters (including spaces).
- President John F. Kennedy's speech have 7618 characters (including spaces).
- President Richard Nixon's speech have 9991 characters (including spaces).

## Number of words:

```
The number of words in President Franklin D. Roosevelt's speech: 1526
The number of words in President John F. Kennedy's speech: 1543
The number of words in President Richard Nixon's speech: 2006
```

- There are 1526 words in President Franklin D. Roosevelt's speech.
- There are 1543 words in President John F. Kennedy's speech.
- There are 2006 words in President Richard Nixon's speech.

## Number of sentences:

```
The number of sentences in President Franklin D. Roosevelt's speech: 68
The number of sentences in President John F. Kennedy's speech: 52
The number of sentences in President Richard Nixon's speech: 68
```

- There are 68 sentences in President Franklin D. Roosevelt's speech.
- There are 52 sentences in President John F. Kennedy's speech.
- There are 68 sentences in President Richard Nixon's speech.

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

Before, removing the stop-words, we have changed all the letters to lowercase and we have removed special characters.

## Word count before the removal of stop-words:

| | Speech | char_count | Processed_Speech | word_count |
|---|---|---|---|---|
| 0 | On each national day of inauguration since 178... | 7571 | on each national day of inauguration since th... | 1334 |
| 1 | Vice President Johnson, Mr. Speaker, Mr. Chief... | 7618 | vice president johnson mr speaker mr chief jus... | 1362 |
| 2 | Mr. Vice President, Mr. Speaker, Mr. Chief Jus... | 9991 | mr vice president mr speaker mr chief justice ... | 1800 |

Before the removal of stop-words,

- President Franklin D. Roosevelt's speech have 1334 words.
- President John F. Kennedy's speech have 1362 words.
- President Richard Nixon's speech have 1800 words.

## Word count after the removal of stop-words:

| | Speech | char_count | Processed_Speech | word_count | stop_count | word_count after removing stopwords |
|---|---|---|---|---|---|---|
| 0 | On each national day of inauguration since 178... | 7571 | national day inauguration since people renewed... | 1334 | 711 | 623 |
| 1 | Vice President Johnson, Mr. Speaker, Mr. Chief... | 7618 | vice president johnson mr speaker mr chief jus... | 1362 | 669 | 693 |
| 2 | Mr. Vice President, Mr. Speaker, Mr. Chief Jus... | 9991 | mr vice president mr speaker mr chief justice ... | 1800 | 969 | 831 |

After the removal of stop-words,

- President Franklin D. Roosevelt's speech have 623 words.
- President John F. Kennedy's speech have 693 words.
- President Richard Nixon's speech have 831 words.

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

## Top 3 words in Roosevelt's speech:

```
The top 3 words in Roosevelt's speech (after removing the stopwords) are:
nation     11
know       10
spirit      9
dtype: int64
```

The top 3 words are,

- nation - 11
- know - 10
- spirit - 9

## Top 3 words in Roosevelt's speech:

```
The top 3 words in Kennedy's speech (after removing the stopwords) are:
let       16
us        12
sides      8
dtype: int64
```

The top 3 words are,

- let - 11
- us - 10
- sides - 9

## Top 3 words in Roosevelt's speech:

```
The top 3 words in Nixon's speech (after removing the stopwords) are:
us       26
let      22
peace    19
dtype: int64
```

The top 3 words are,

- us - 26
- let - 22
- peace - 19

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

## Word cloud of Roosevelt's speech:

## Word cloud of Kennedy's  speech:

## Word cloud of Nixon's speech: