PROJECT
ON
MACHINE LEARNING

By SHAJIL FERNANDEZ

01-10-2023

Table of Contents:

Problem 1

Problem 2

List of Figures:

List of Tables:

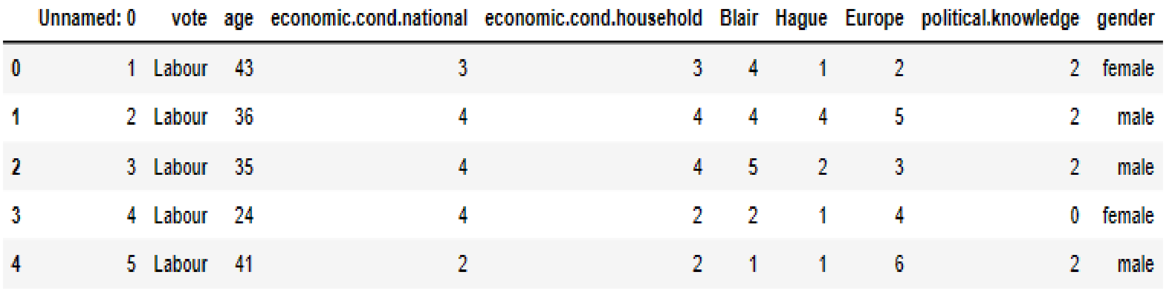
Problem 1

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.

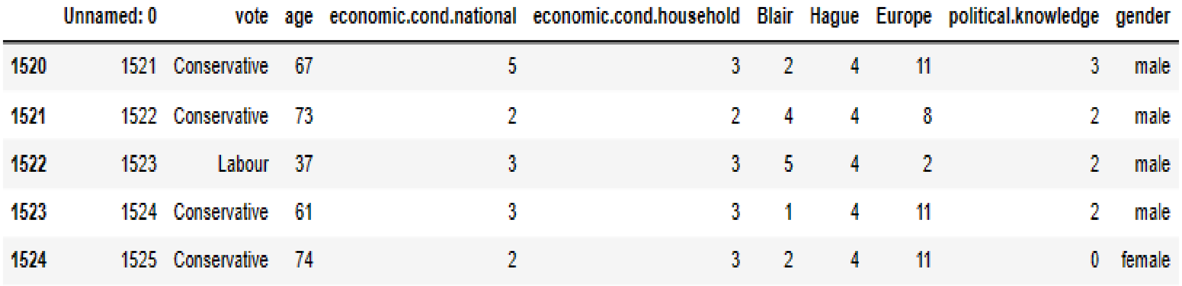
Ans:

- **Table 1: Top 5 rows of the dataset**



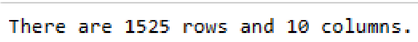
	Unnamed: 0	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	1	Labour	43	3	3	4	1	2	2	female
1	2	Labour	36	4	4	4	4	5	2	male
2	3	Labour	35	4	4	5	2	3	2	male
3	4	Labour	24	4	2	2	1	4	0	female
4	5	Labour	41	2	2	1	1	6	2	male

- **Table 2: Bottom 5 rows of the dataset**



	Unnamed: 0	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
1520	1521	Conservative	67	5	3	2	4	11	3	male
1521	1522	Conservative	73	2	2	4	4	8	2	male
1522	1523	Labour	37	3	3	5	4	2	2	male
1523	1524	Conservative	61	3	3	1	4	11	2	male
1524	1525	Conservative	74	2	3	2	4	11	0	female

- There are 1525 rows and 10 columns in the given data.

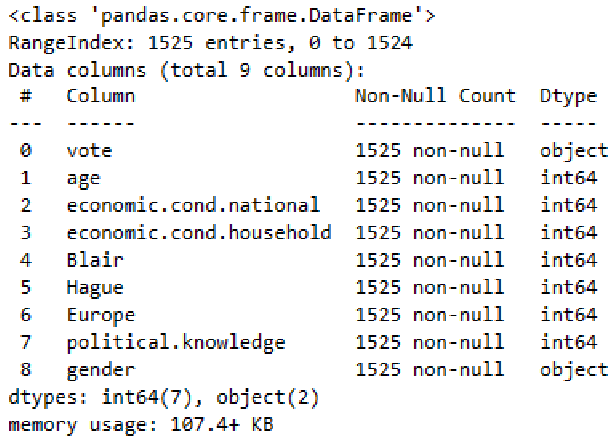


```
There are 1525 rows and 10 columns.
```

After deleting the Index column, there are 1525 rows and 9 columns in the given data.

- **Info**:

  Table 3: Basic info of the dataset



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

  Observations:

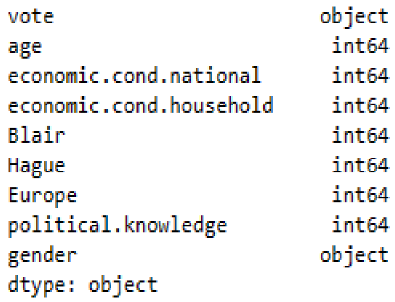
  - There are 1525 rows and 9 columns.

  - There are 7 int64 and 2 object datatypes.

  - There are no missing values in the given data.

- **Datatypes**:

  Table 4: Datatype of the respective columns



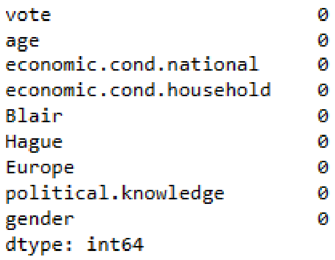
```
vote                       object
age                         int64
economic.cond.national      int64
economic.cond.household     int64
Blair                       int64
Hague                       int64
Europe                      int64
political.knowledge         int64
gender                  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

- **Duplicates**:

  There are **8 duplicate rows** in the given data. So, we can go ahead and delete them. After deleting the duplicates, there are 1517 rows and 9 columns in the given data.

- **Null values**:

Table 5: Table showing missing value info in the dataset

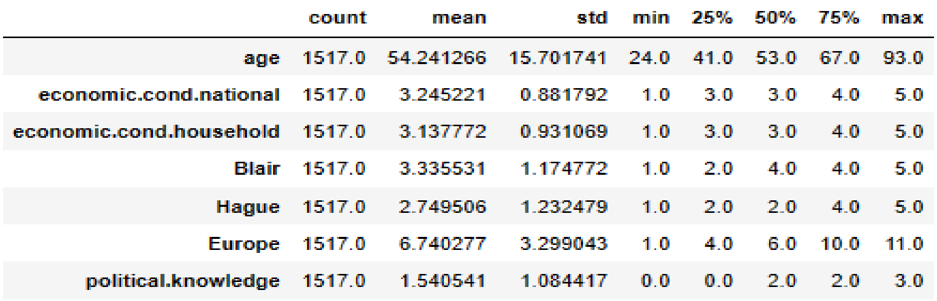


```
vote                       0
age                        0
economic.cond.national     0
economic.cond.household    0
Blair                      0
Hague                      0
Europe                     0
political.knowledge        0
gender                     0
dtype: int64
```

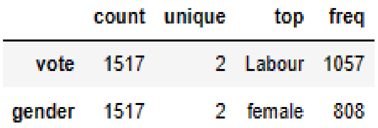
There are **no missing values** in the given dataset.

- **Statistical summary of numerical and categorical data:**

Table 6: Table showing 5-point summary



	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



	count	unique	top	freq
vote	1517	2	Labour	1057
gender	1517	2	female	808

Observations:

- Average age of voters is 54 years.

- About 75% of the voters have political knowledge of 2 (on 0-3 scale).

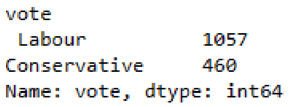
- Of 1517 voters, 808 are females.

- Majority is the Labour party when compared with Conservative party.

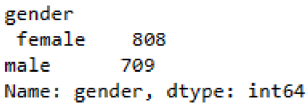
- About 50% of the voters have current household economic conditions of 3 (on 1-5 scale).

- **Unique values:**

Table 7: Table showing unique values of the categorical data



```
vote
 Labour          1057
Conservative     460
Name: vote, dtype: int64
```



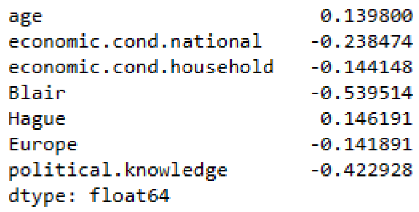
```
gender
 female     808
male       709
Name: gender, dtype: int64
```

Observations:

- Labour party is majority when compared with Conservative party i.e., 1057 of 1517 prefer Labour party.

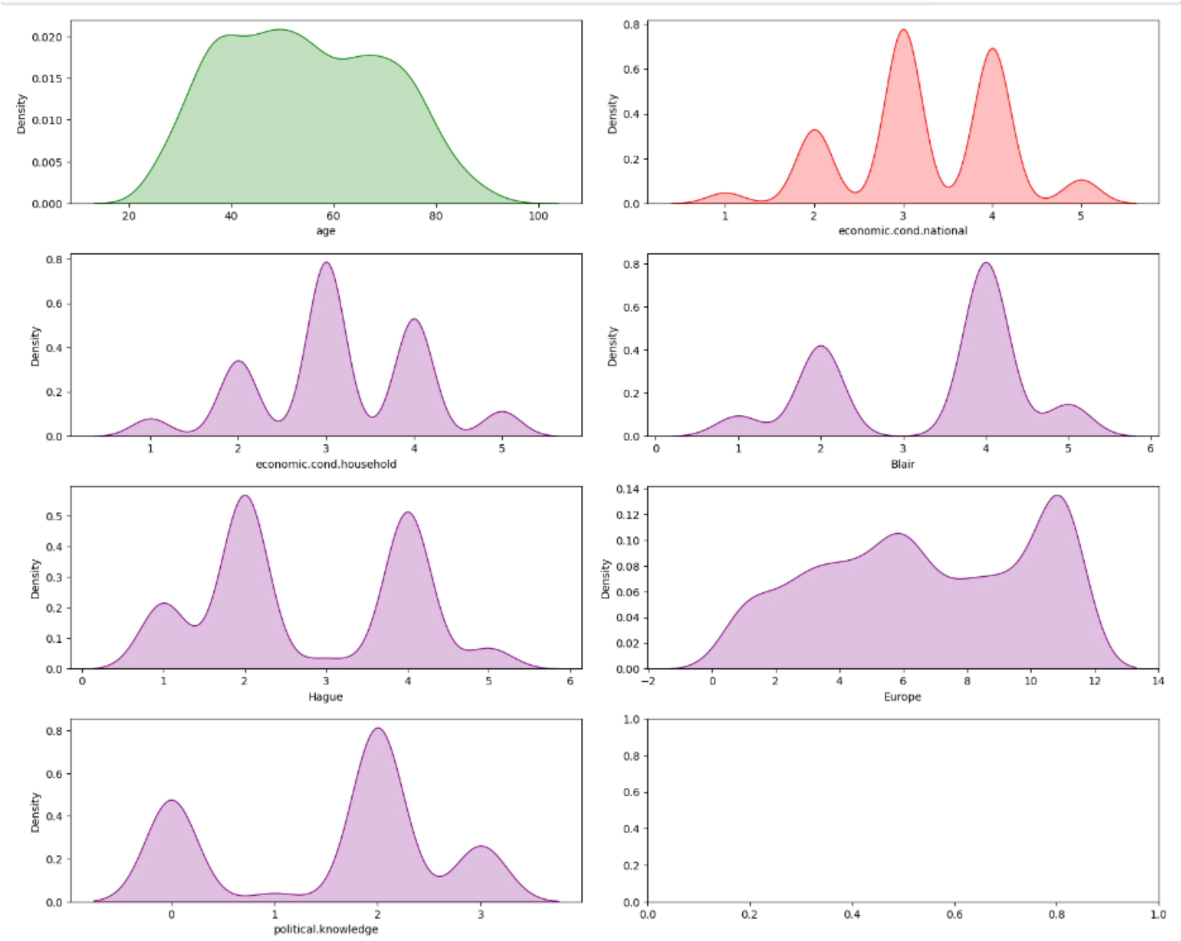
- Female voters are more when compared with Male voters.

- **<u>Skewness</u>:**



```
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
```

Figure 1: Kdeplot showing the skewness of the variables



Observations:

- The distribution of voters **Age** is positive or right-skewed, with skewness of 0.140.

- The distribution of **‘current national economic conditions**’ is negative or left-skewed, with skewness of -0.238.
- The distribution of **‘current household economic conditions**’ is negative or left-skewed, with skewness of -0.144.
- The distributions of ‘Blair’, ‘Europe’ and ‘Political knowledge’ are also negatively or left-skewed, with skewness of -0.540, -0.142 and -0.423 as follows.

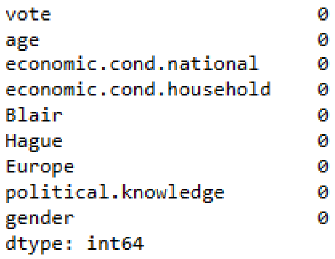
- The distribution of ‘Hague’ is positive or right-skewed, with skewness of 0.146.

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**.**

Ans:

- **Null values:**

Table 8: Table showing missing value info in the dataset

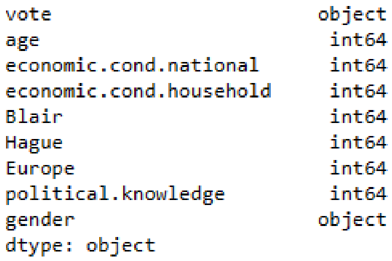


```
vote                       0
age                        0
economic.cond.national     0
economic.cond.household    0
Blair                      0
Hague                      0
Europe                     0
political.knowledge        0
gender                     0
dtype: int64
```

There are **no missing values** in the given dataset.

- **Datatypes:**

Table 9: Datatype of the respective columns



```
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
```

Observations:

- There are 7 int64 and 2 object datatypes.

- **Shape**:

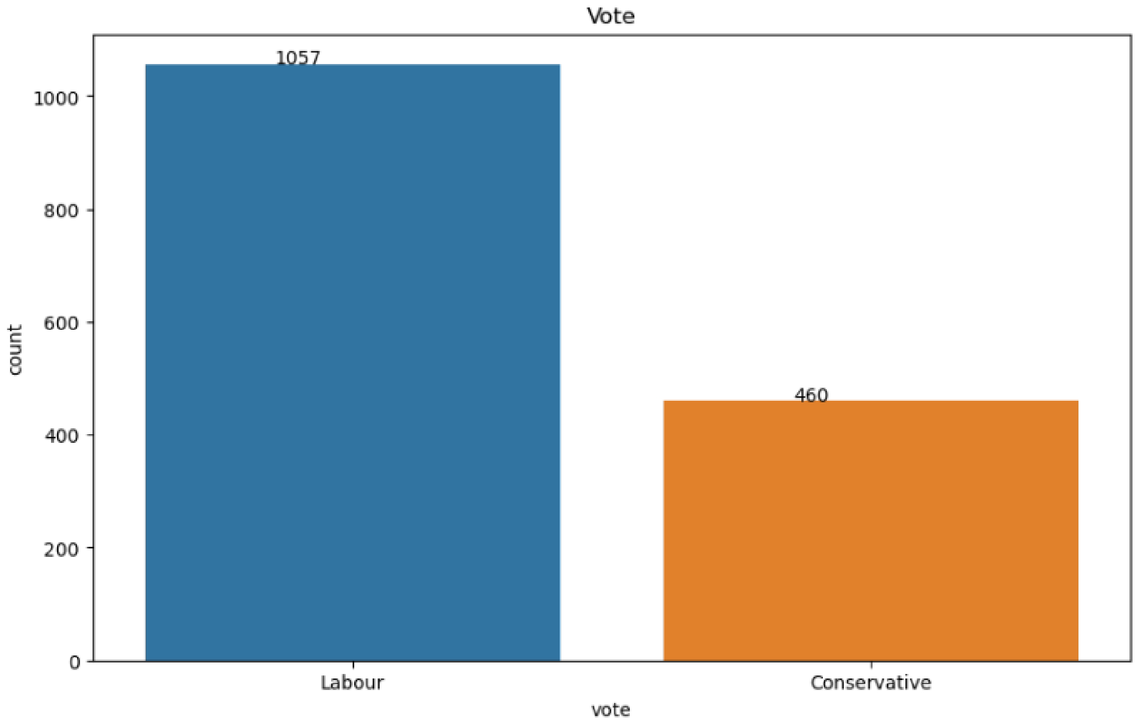
  There are 1517 rows and 9 columns in the given data.

```
There are 1517 rows and 9 columns.
```

- **Univariate Analysis:**

a) Vote:

Figure 2: Countplot (vote)



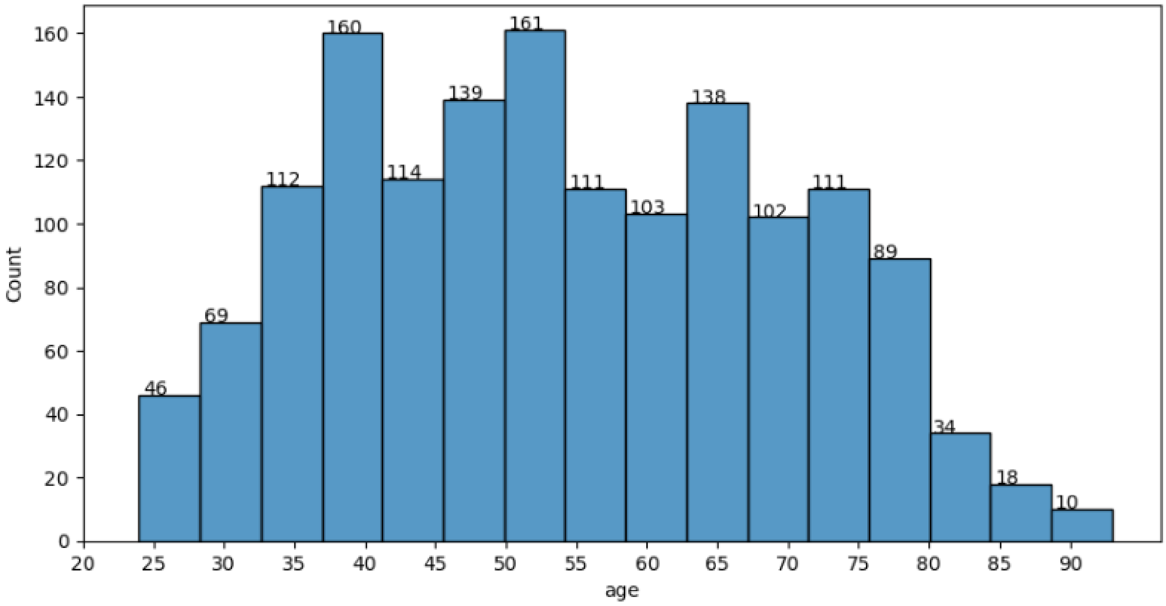
Observations

- 1057 of 1517 voters chose Labour party over Conservative party.

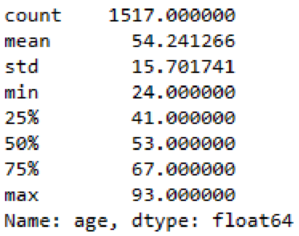
- 460 voters chose Conservative party.

b) Age:

Figure 3: Histplot (age)



Statistical summary of the variable



```
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
```

Observations

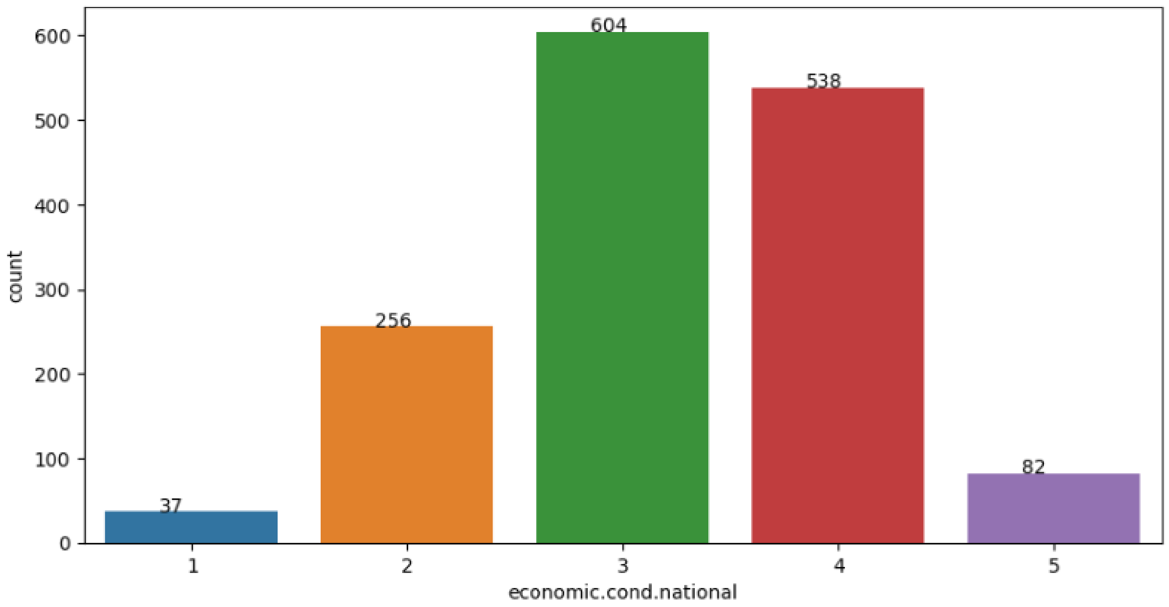
- Average age of voters as per the data is 54 years.

- Minimum age of voter is 24 years and maximum age is 93 years.

- About 75% of data has voters below 67 years.

c) Current national economic conditions:

Figure 4: Countplot (current national economic conditions)



Observations

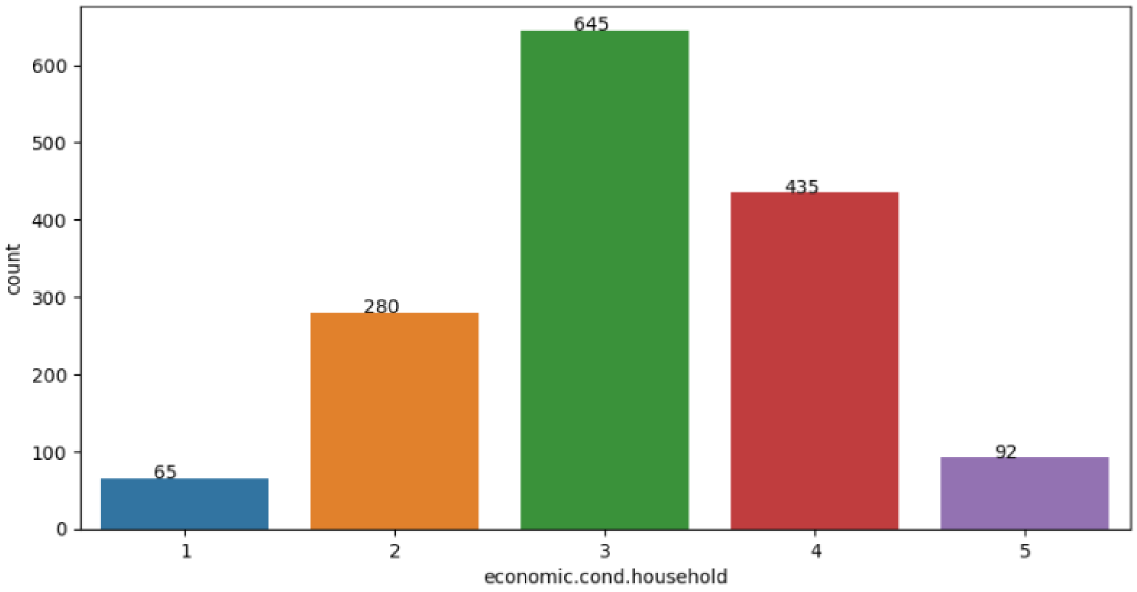
- 604 of 1517 voters rated 3 for the current national economic conditions.

- Only 37 voters gave 1 rating for the current national economic conditions.

- 82 voters rated 5 on scale 5 for the current national economic conditions.

d) Current household economic conditions:

Figure 5: Countplot (current household economic conditions)



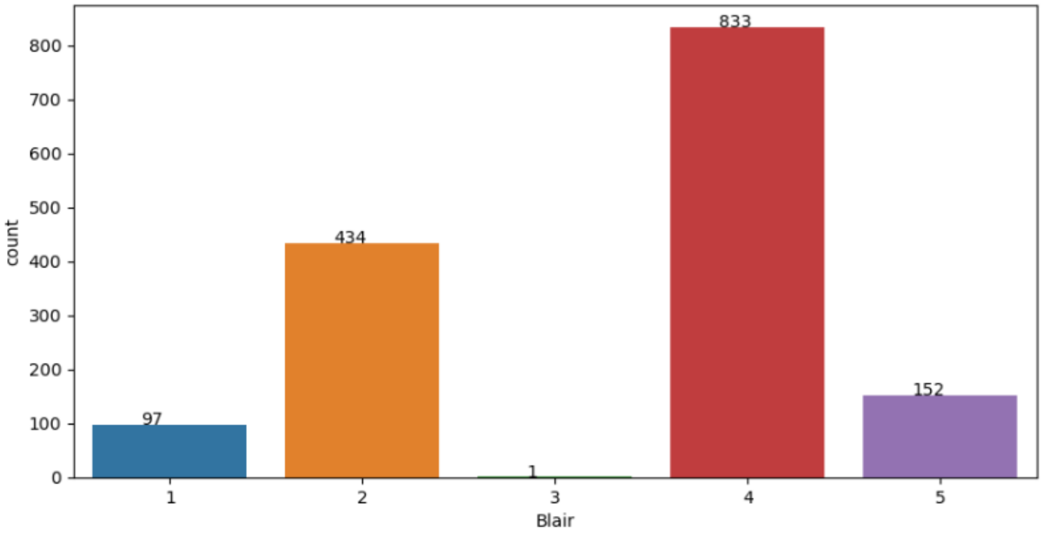
Observations

- For the current household economic conditions, 65 voters rated 1 and 92 voters rated 5 on the scale of 5.

- Many voters opted 3 rating for the household economic conditions.

e) Blair:

Figure 6: Countplot (Blair)



Observations

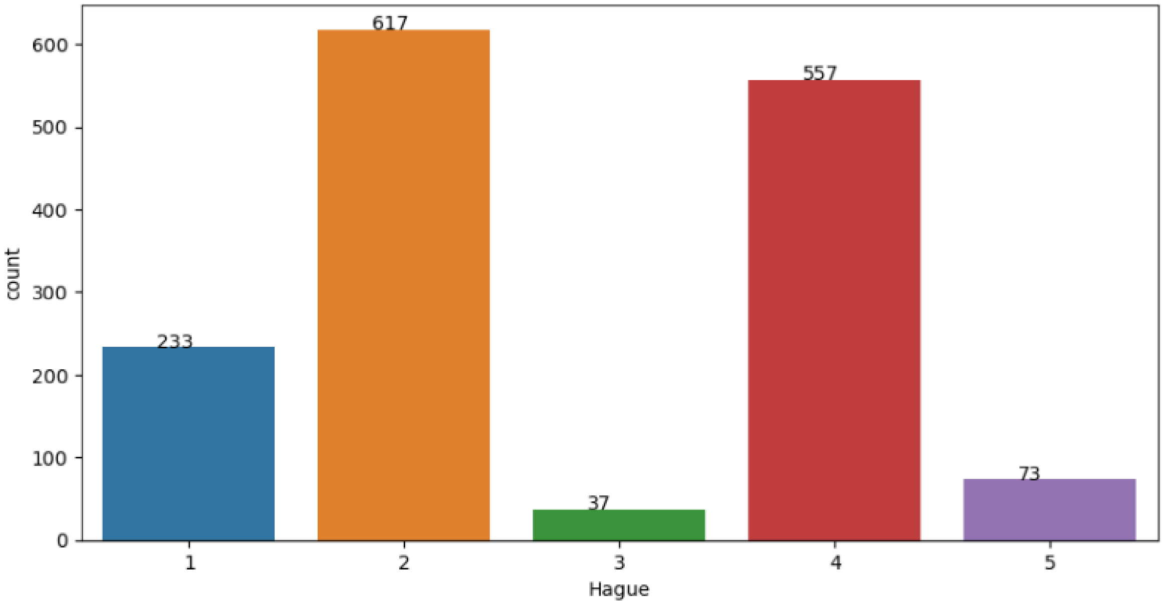
- 97 voters rated 1 for the Labour party leader.

- 152 voters have given a rating of 5 for the Labour party leader.

- Majority of the voters i.e., 833 of them rated 4 for the leader of the Labour party.

f) Hague:

Figure 7: Countplot (Hague)



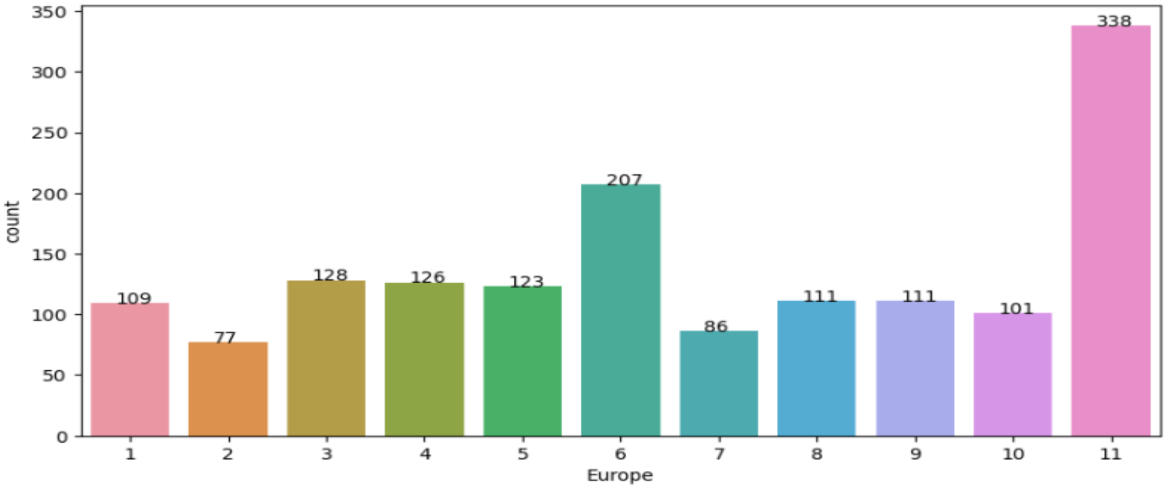
Observations

- Of 1517 voter, 73 of them rated 5 for the leader of the Conservative party.

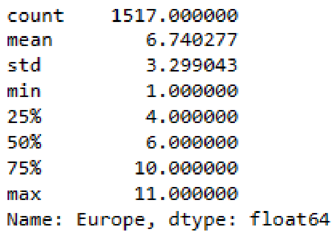
- Majority of the voters i.e., 617 of them rated 2 for the Conservative party leader and 233 voters gave 1 rating.

g) Europe:

Figure 8: Countplot (Europe)



Statistical summary of the variable



```
count    1517.000000
mean        6.740277
std         3.299043
min         1.000000
25%         4.000000
50%         6.000000
75%        10.000000
max        11.000000
Name: Europe, dtype: float64
```

Observations

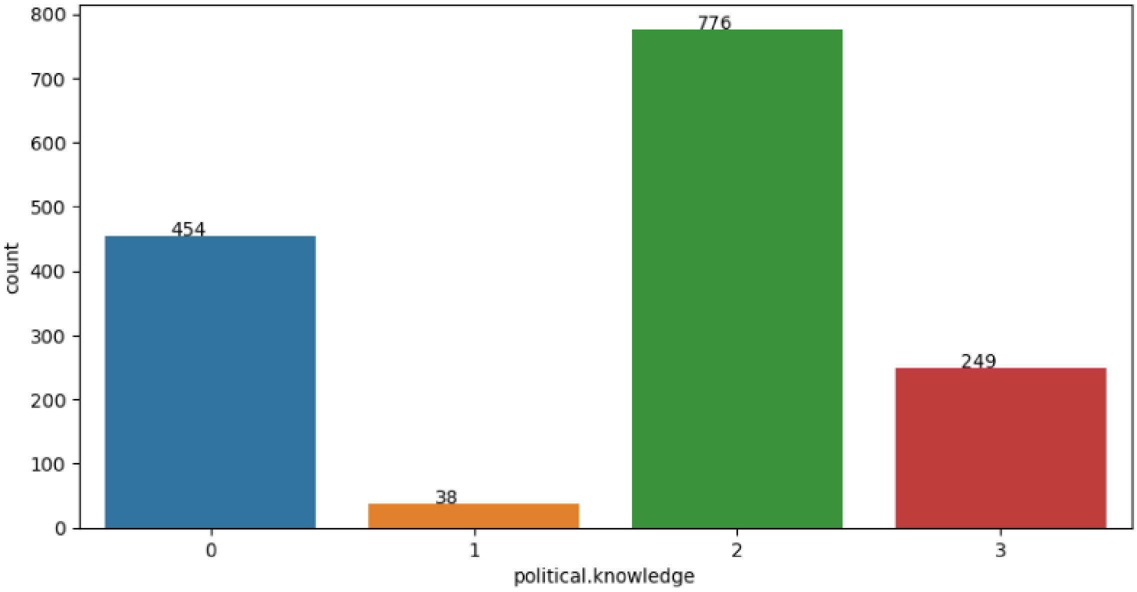
- About 338 voters rated 11 which means they oppose the European integration.

- 75% of voters rated below 10.

- 50% of voters rated below 6.

h) **Political knowledge**:

Figure 9: Countplot (political knowledge)



Observations

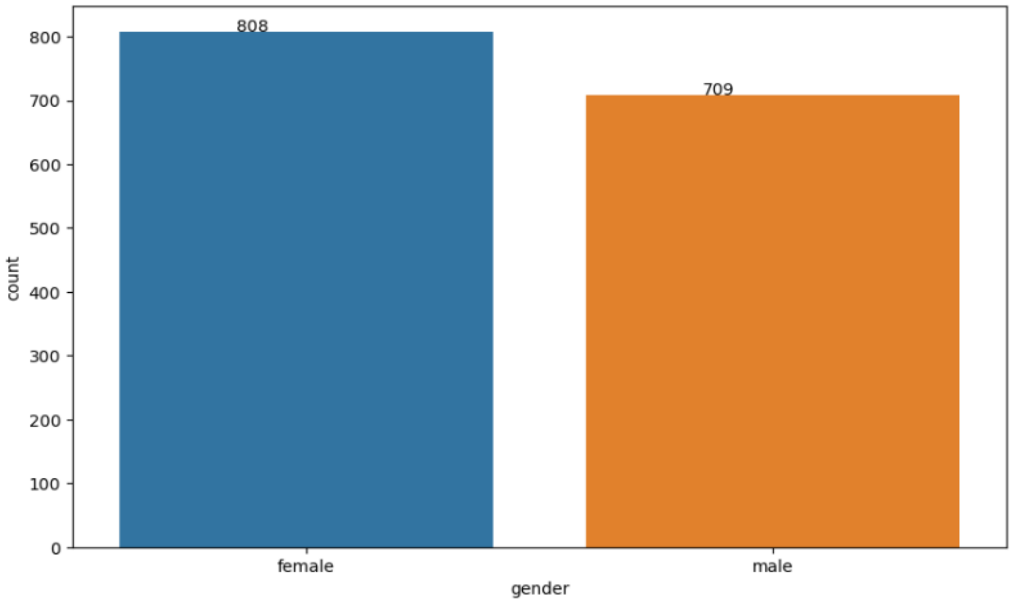
- 776 voters have opted 2 on 3 scale about the political knowledge.

- 454 voters have 0 political knowledge.

- 249 voters have in depth political knowledge.

i) Gender:

Figure 10: Countplot (Gender)



Observations

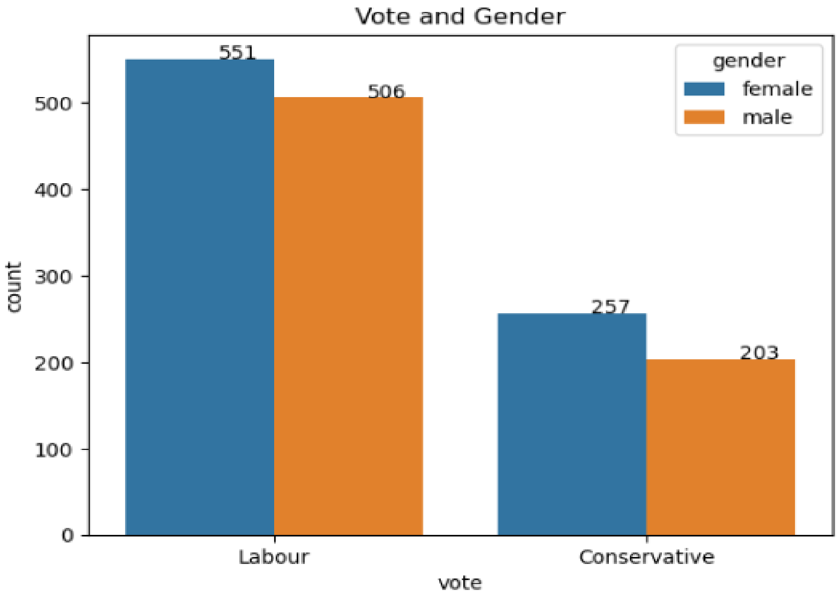
- Female voters are more, when compared with Male voters.

- 808 voters are female and 709 voters are male.

- **<u>Bivariate Analysis and Multivariate Analysis:</u>**

a) Vote and Gender:

Figure 11: Countplot (Vote and Gender)



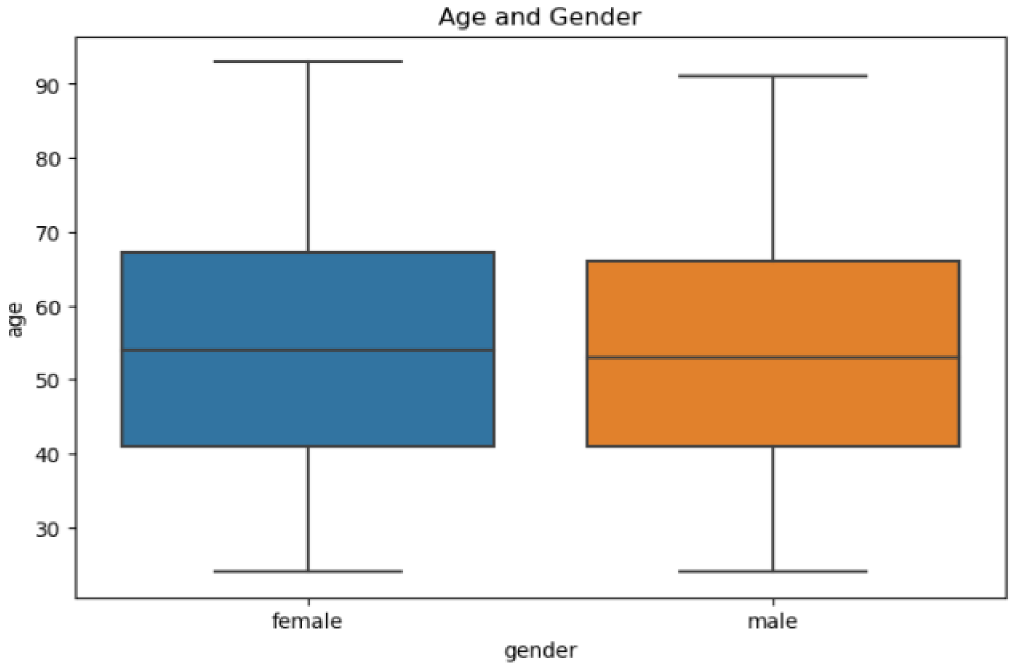
Observations

- Majority of them have opted Labour party (about 1057) over Conservative party, of which, female voters are more than male voters.

- 460 voters prefer Conservative party, of which, 257 are females and 203 are males.

b) Gender and Age:

Figure 12: Boxplot (Gender and Age)



Observations

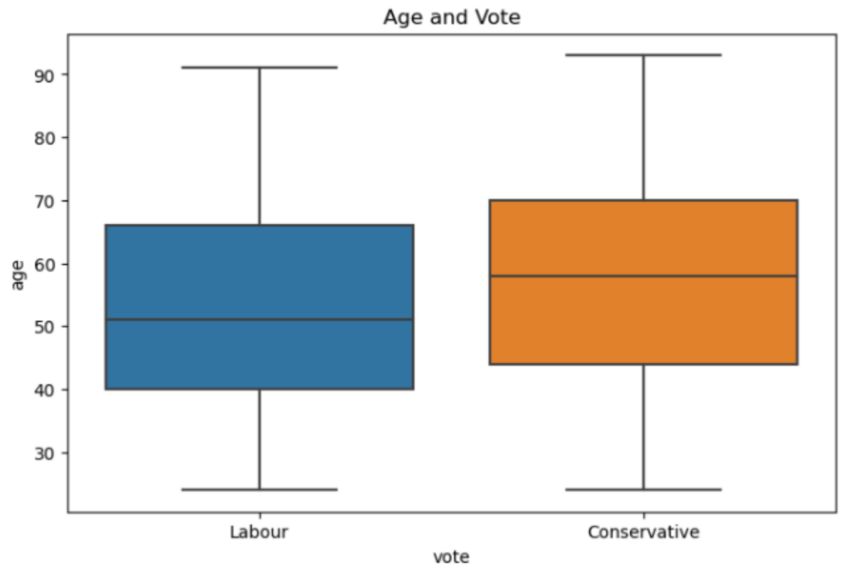
- Median age of female voters is 54 years and male voters is 53 years.

- Female voters age ranges from 24 to 93 years.

- Male voters age ranges from 24 to 91 years.

c) Vote and Age:

Figure 13: Boxplot (Vote and Age)



Observations

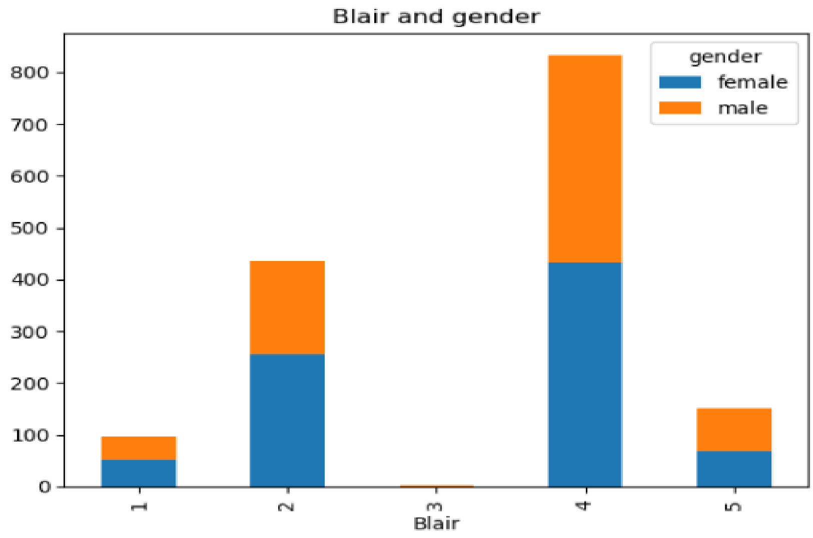
- Median age of voters who prefer Labour party is 51 years and those who prefer Conservative party is 58 years.

- About 75% of Labour party voters fall under 66 years of age.

- About 75% of Conservative party voters fall under 70 years of age.

d) Blair and Gender:

Figure 14: Crosstab (Blair and Gender)



Observations

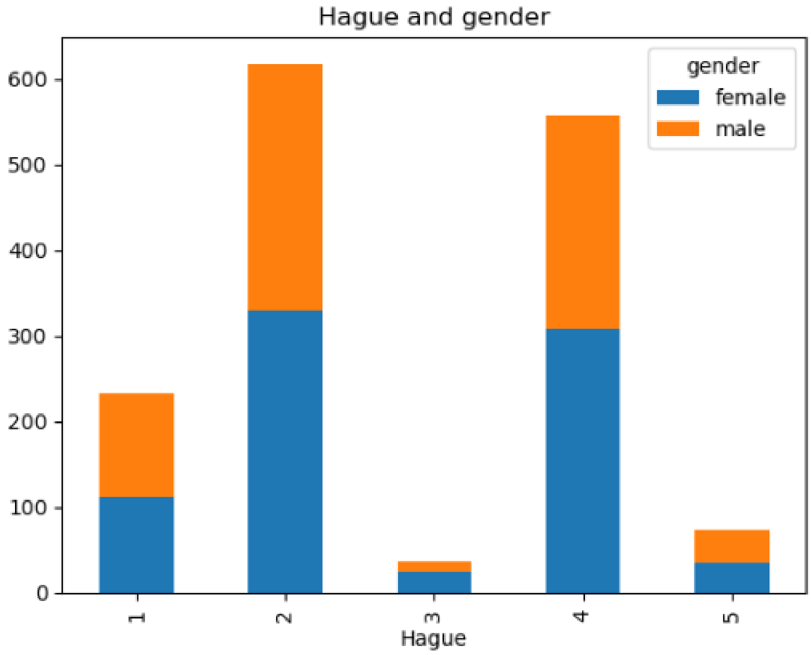
- From the above graph, we can observe that most of them have given 4 rating to the Labour party leader, of which 432 are females and 401 are males.

- Only 1 voter has given 3 rating.

- 83 male voters and 69 female voters have given 5 rating on scale of 5.

e) Hague and Gender:

Figure 15: Crosstab (Hague and Gender)



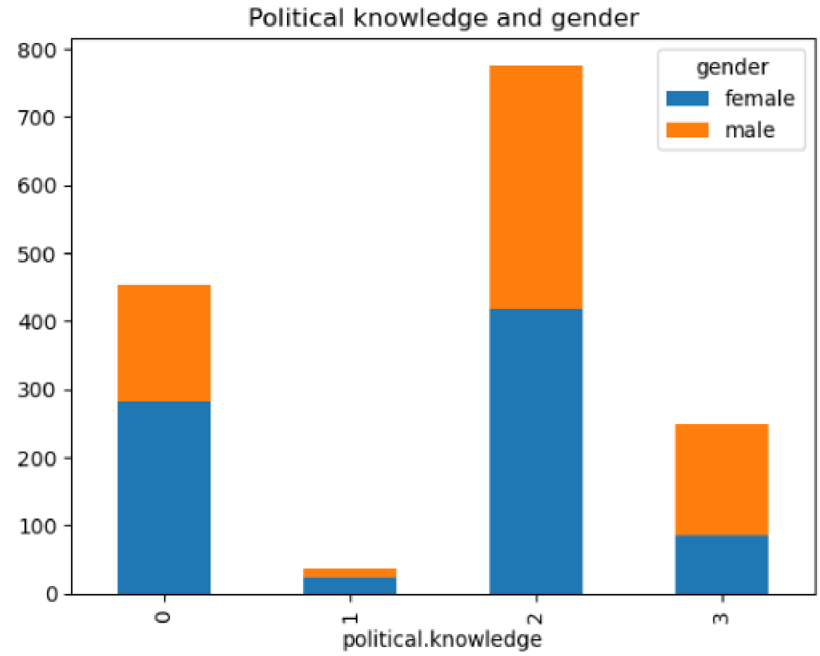
Observations

- We can observe that most of them have given 2 rating to the Conservative party leader, of which 329 are females and 288 are males.

- 309 female voters and 248 male voters have given 4 rating to the leader of Conservative party, which is less when compared with rating given to the Labour party leader.

f) Political knowledge and Gender:

Figure 16: Crosstab (Political knowledge and Gender)



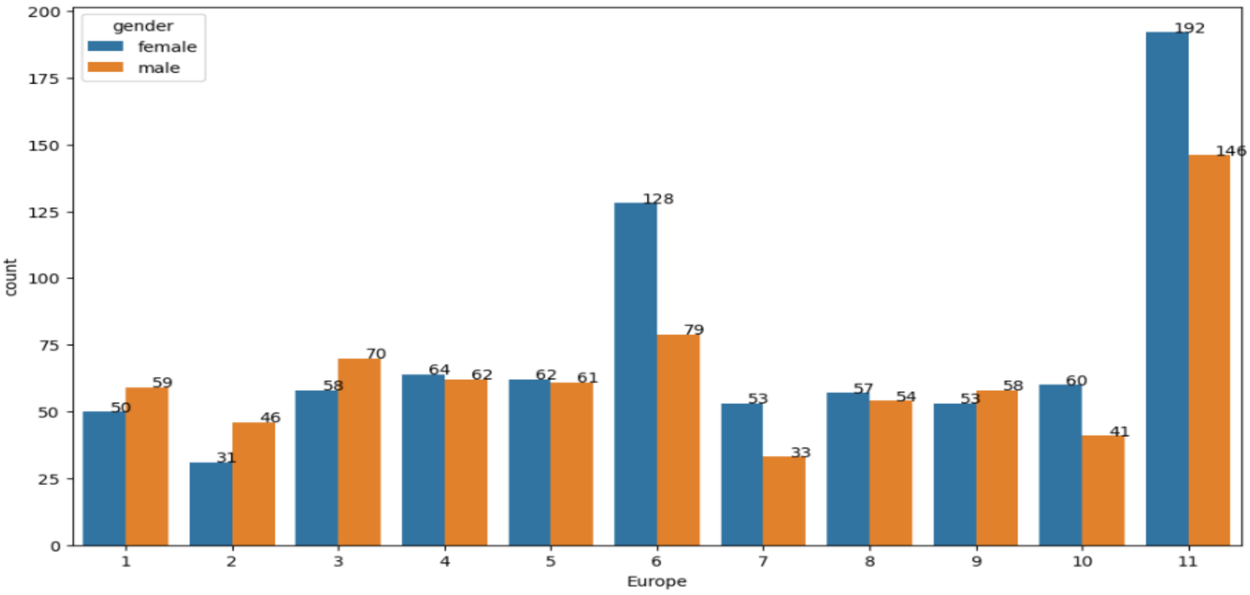
Observations

- 454 of 1517 voters have zero knowledge in politics, of which 281 are females and 173 are males.

- 164 male voters and 85 female voters have good knowledge in politics.

g) **Gender and Europe**:

Figure 17: Countplot (Gender and Europe)



Observations

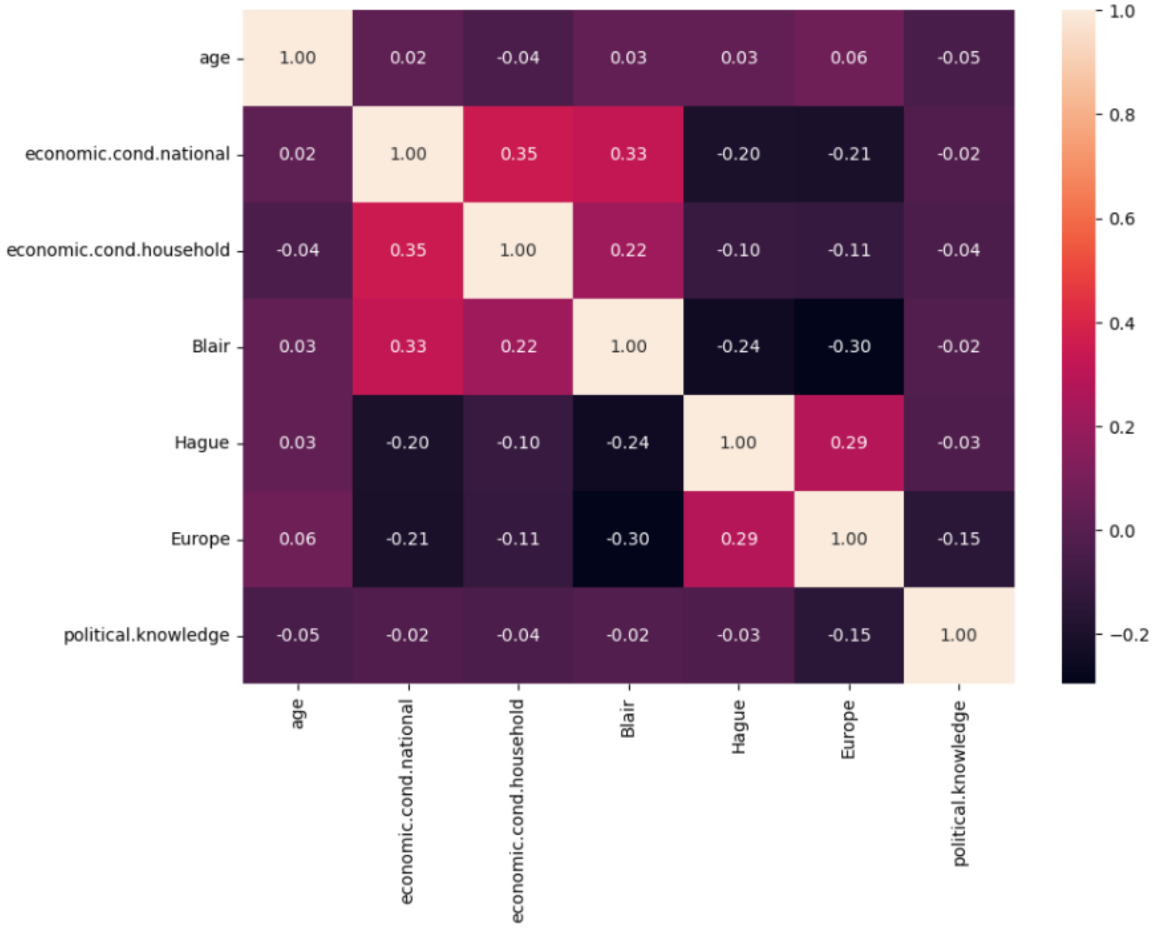
- Most of the voters oppose European integration, i.e., about 192 female voters and 146 male voters have given 11 on scale 11.

- We can also see an increase in the point 6, of which 128 are females and 79 are males.

- 50 female voters and 59 male voters have 1 point, i.e., on the favour of European integration.

h) Correlation:

Figure 18: Heatmap to check correlation between variables



Observations

- Current national economic conditions and Current household economic conditions are positively correlated with the correlation of 0.35.

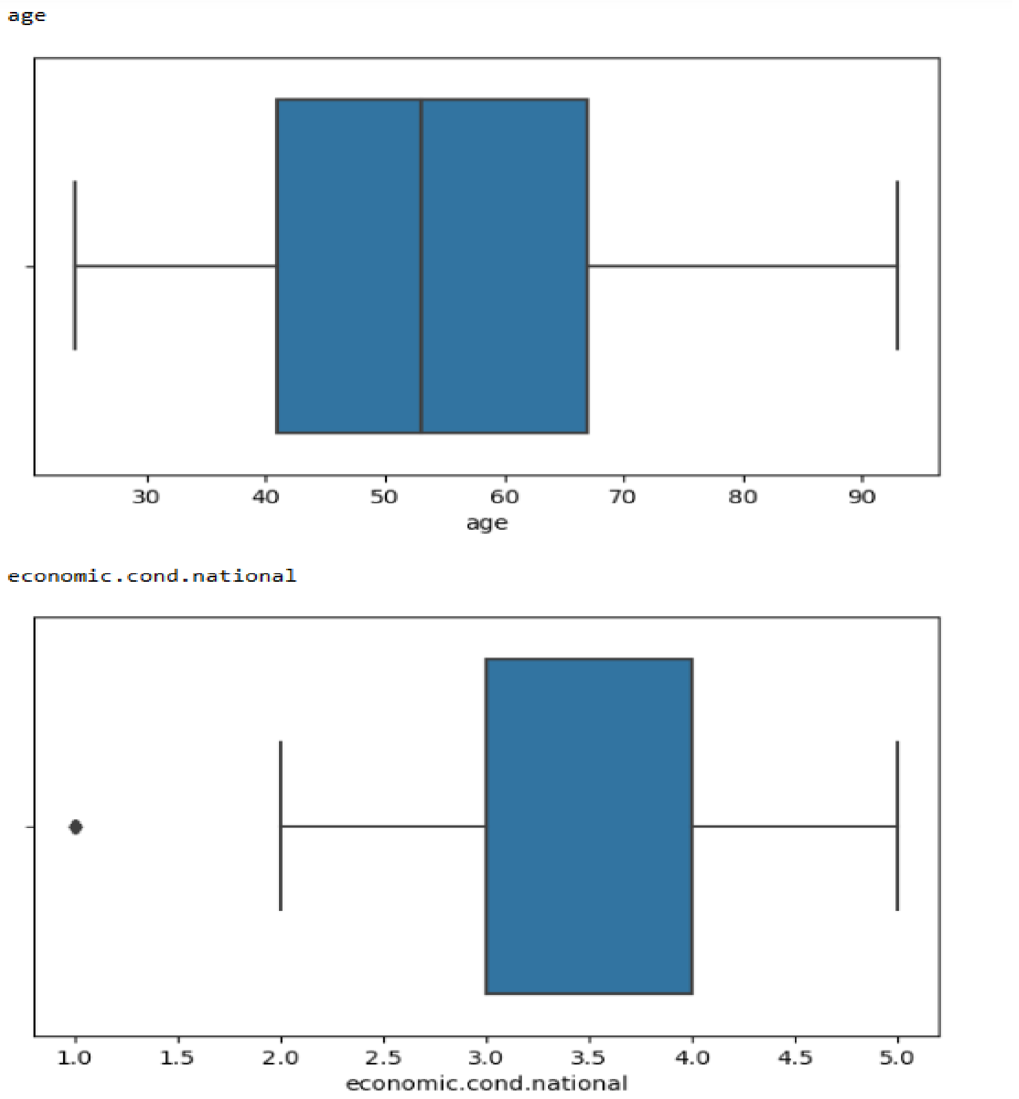
- Current national economic conditions and Blair are positively correlated with the correlation of 0.33.

- Europe and Blair are negatively correlated with the correlation of -0.30.

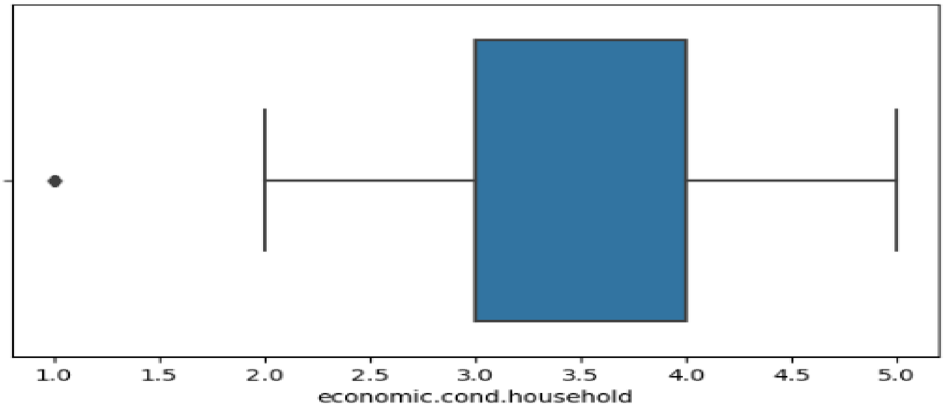
- Europe and Current national economic conditions are negatively correlated with the correlation of -0.21.

- **<u>Outliers</u>:**

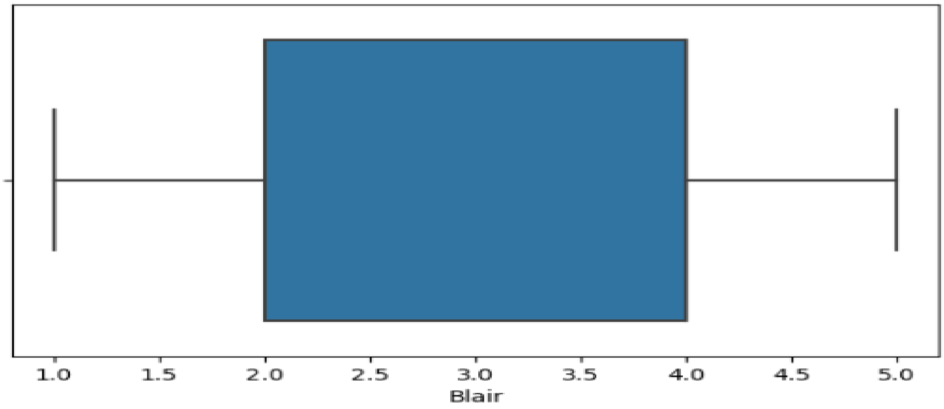
Figure 19: Boxplot to check outliers



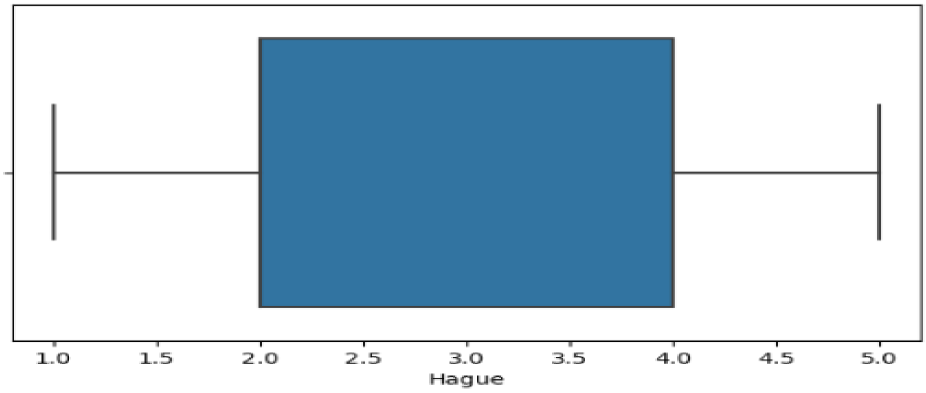
economic.cond.household



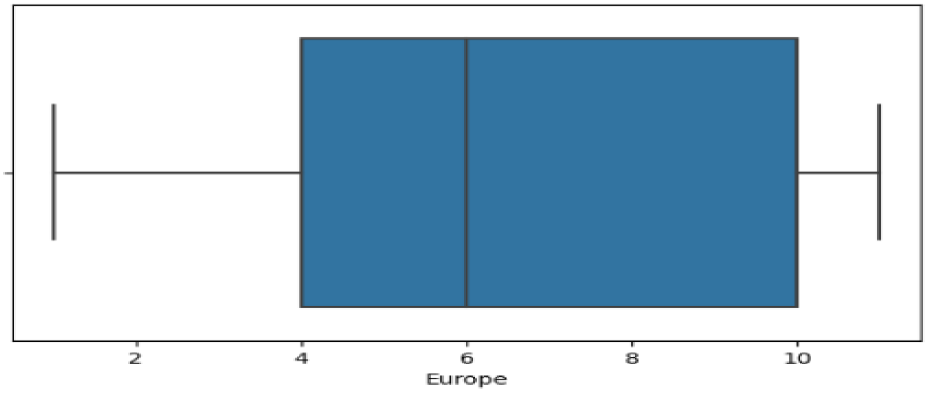
Blair

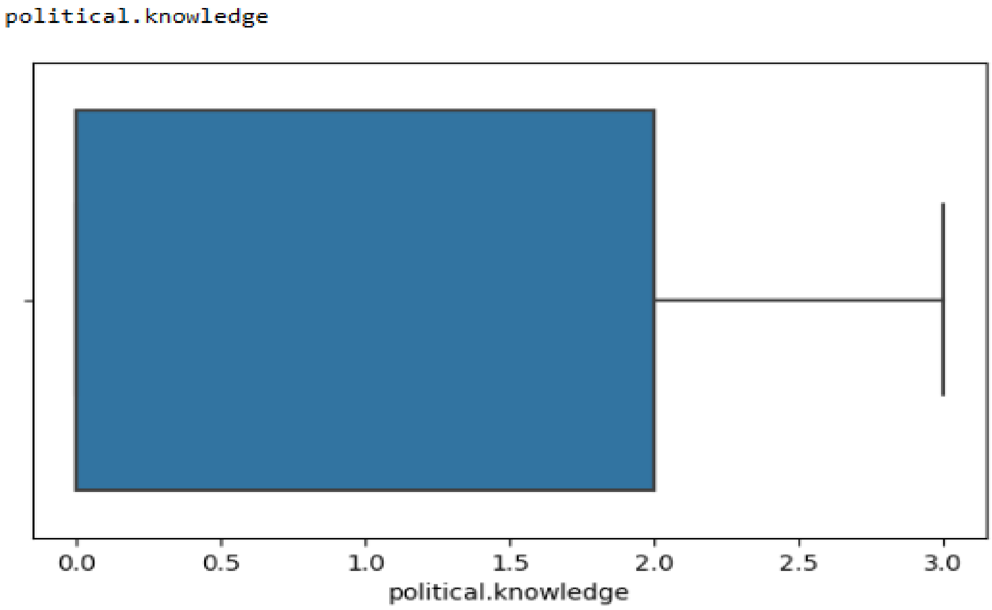


Hague



Europe





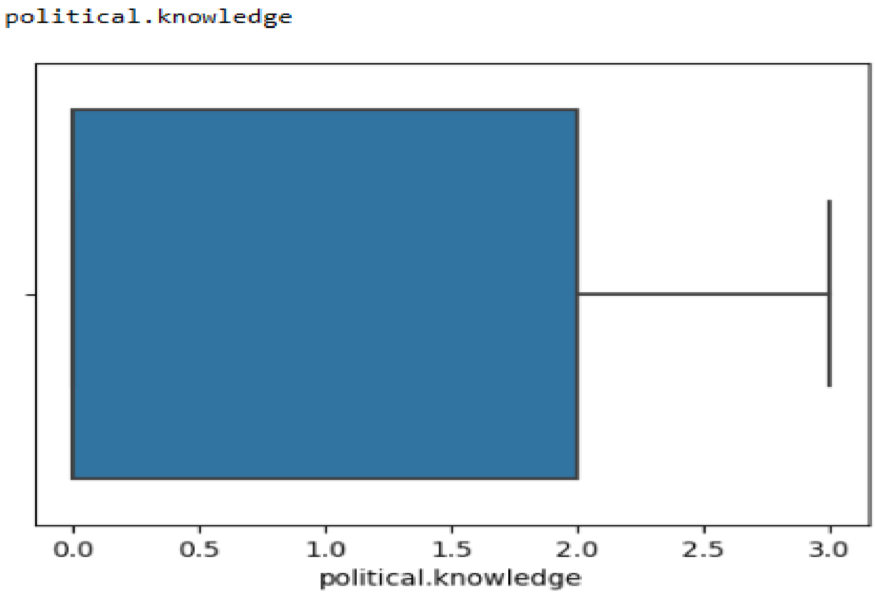
Inference

- We can observe that there are outliers for two variables ‘current national economic conditions’ and ‘current household economic conditions’ but, since these variables are of ordinal datatypes, we should not treat the outliers as the outliers are treated only for the continuous variables.

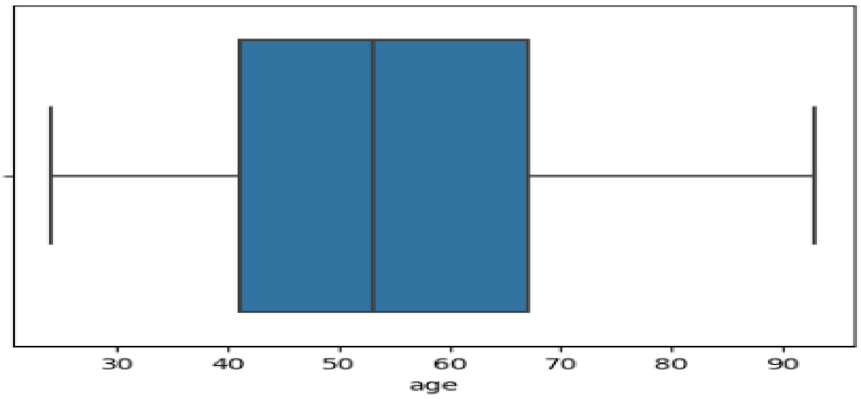
However, we are showing the treatment by creating a new data frame (df_num) and by using the IQR (95th percentile) method.

Treating outliers:

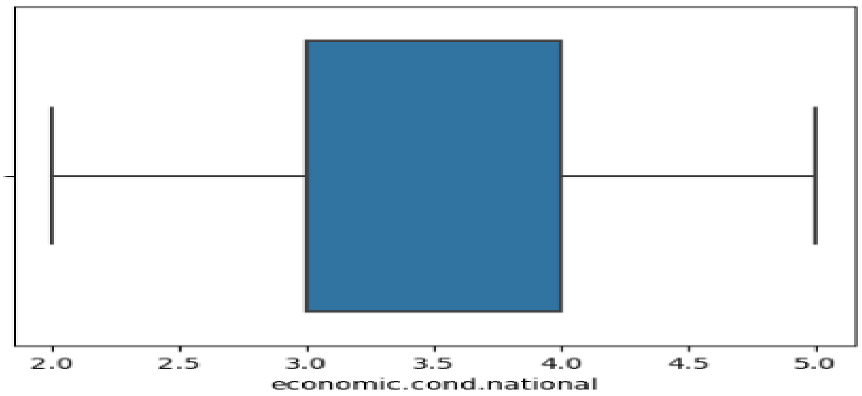
Figure 20: Boxplot after treating outliers



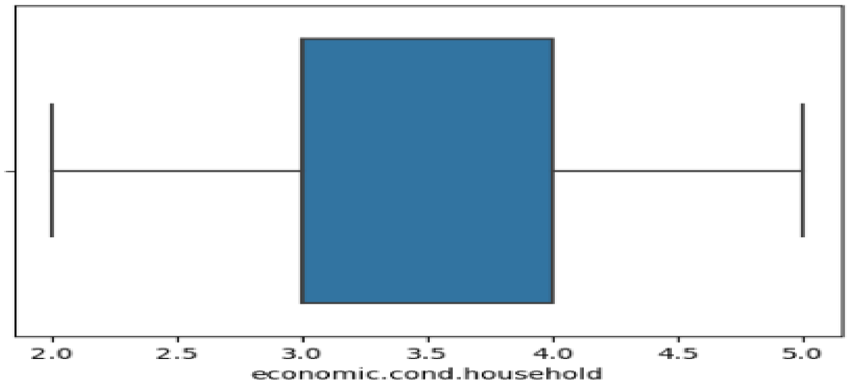
```
age
```



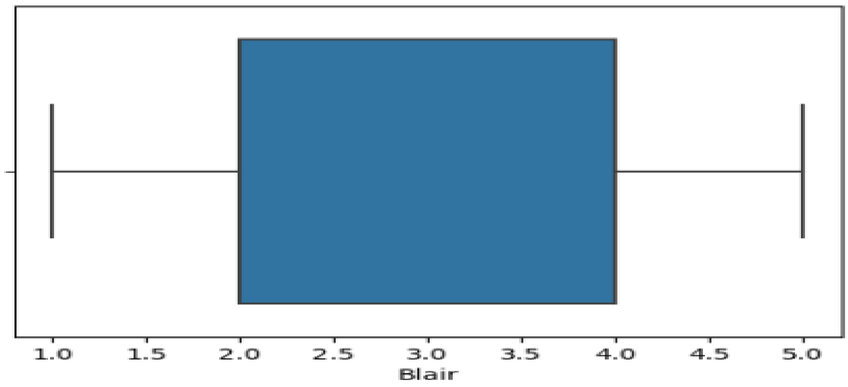
```
economic.cond.national
```

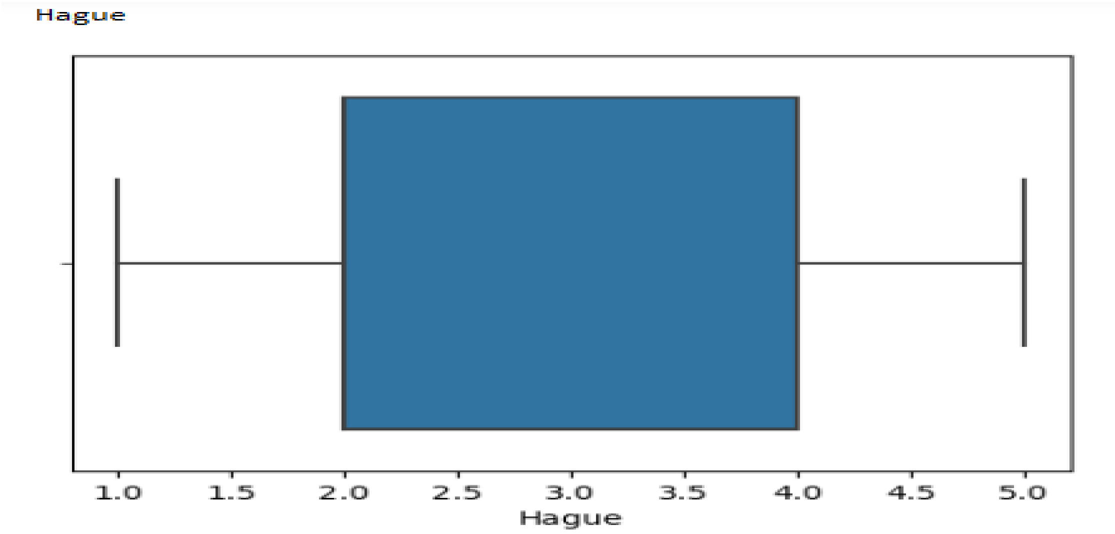


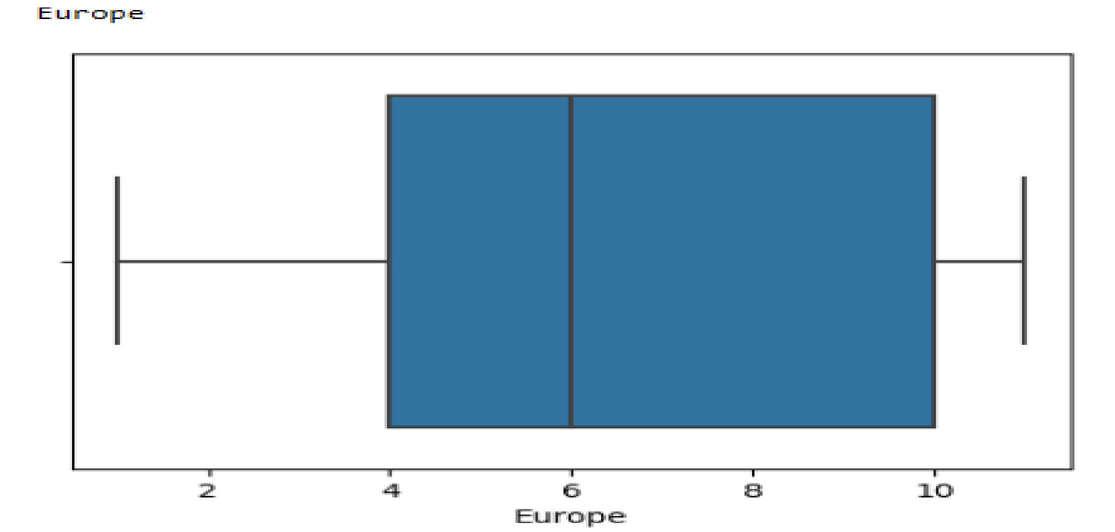
```
economic.cond.household
```



```
Blair
```







There were outliers only for two variables, hence, we have treated the same by using 95th percentile method which covers 95% of the data and excludes extreme outliers. However, in this case all the data is covered as there were no extreme outliers.

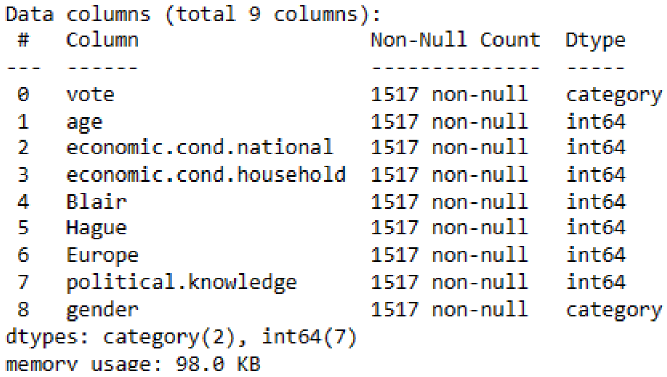
1.3 Encode the data (having string values) for Modelling. Is Scaling necessary here or not? (2 pts), Data Split: Split the data into train and test (70:30) (2 pts). The learner is expected to check and comment about the difference in scale of different features on the bases of appropriate measure for 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.

Ans:

Encoding data

- Initially, we converted the object datatype (2 variables) into categorical data.

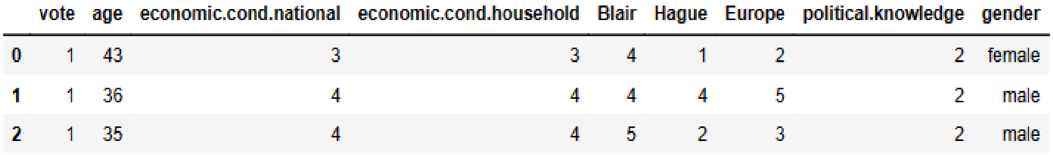
Table 10: Table showing datatype of respective variables



```
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype   
---  ------                   --------------  -----   
 0   vote                     1517 non-null   category
 1   age                      1517 non-null   int64   
 2   economic.cond.national   1517 non-null   int64   
 3   economic.cond.household  1517 non-null   int64   
 4   Blair                    1517 non-null   int64   
 5   Hague                    1517 non-null   int64   
 6   Europe                   1517 non-null   int64   
 7   political.knowledge      1517 non-null   int64   
 8   gender                   1517 non-null   category
dtypes: category(2), int64(7)
memory usage: 98.0 KB
```

- Applying Label Encoder to the target class, where '1' represents 'Labour' party and '0' represents Conservative party.

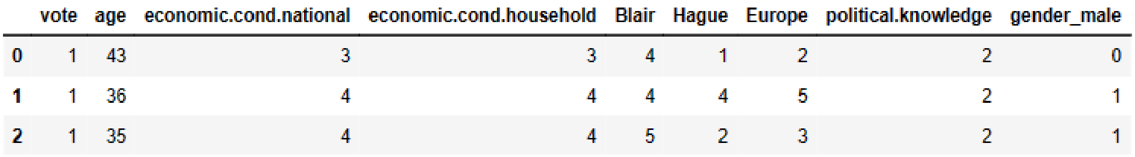
Table 11: Table showing LabelEncoder applied



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

- Creating the dummy variable from already converted categorical data i.e., gender column.

Table 12: Table showing dummy variable (gender column)

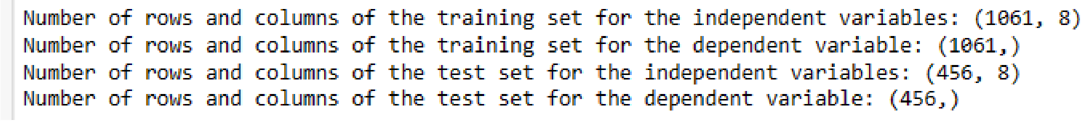


	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
0	1	43	3	3	4	1	2	2	0
1	1	36	4	4	4	4	5	2	1
2	1	35	4	4	5	2	3	2	1

- **Train and Test Split:**

Assigning the independent variables to 'X' and dependent variables to 'y' and splitting the data into Train and Test at 70:30 ratio.

Table 13: Table showing train test split



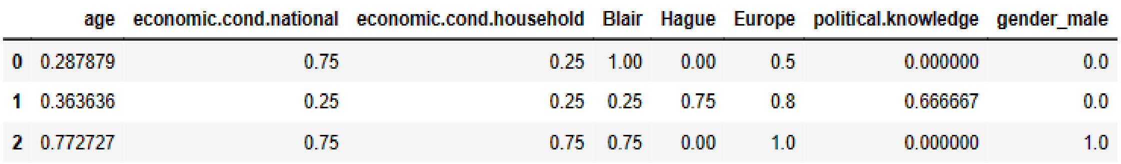
```
Number of rows and columns of the training set for the independent variables: (1061, 8)
Number of rows and columns of the training set for the dependent variable: (1061,)
Number of rows and columns of the test set for the independent variables: (456, 8)
Number of rows and columns of the test set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

- **Scaling data:**

- We need to scale the data as we have continuous and ordinal variables with different measures.
- Min Max method can be used to scale the data as we also have nominal variables in the data. Min Max method does not affect the nominal variables, as the values would not change even after the calculation.

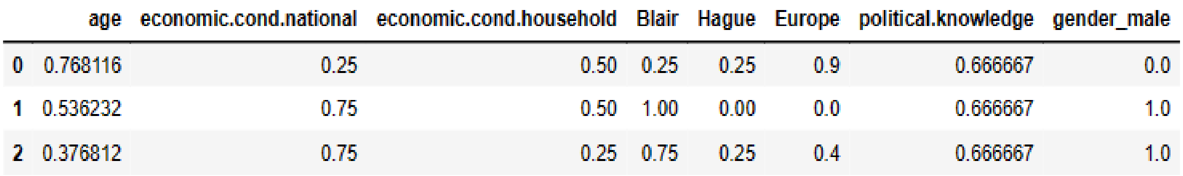
Table 14: Table showing scaled data (train and test)

Train data



	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
0	0.287879	0.75	0.25	1.00	0.00	0.5	0.000000	0.0
1	0.363636	0.25	0.25	0.25	0.75	0.8	0.666667	0.0
2	0.772727	0.75	0.75	0.75	0.00	1.0	0.000000	1.0

Test data



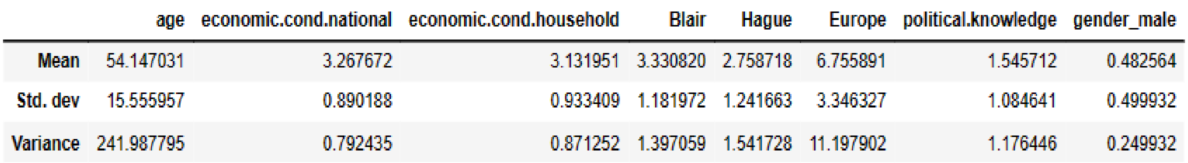
	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
0	0.768116	0.25	0.50	0.25	0.25	0.9	0.666667	0.0
1	0.536232	0.75	0.50	1.00	0.00	0.0	0.666667	1.0
2	0.376812	0.75	0.25	0.75	0.25	0.4	0.666667	1.0

- **Original data vs Scaled data (std, variance)**

Original data

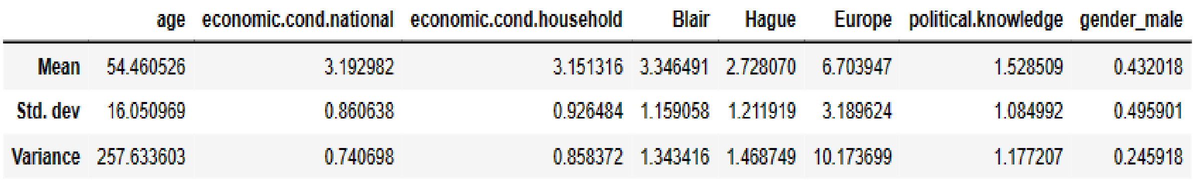
Table 15: Table showing standard deviation and variance of original data

Train data



	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
Mean	54.147031	3.267672	3.131951	3.330820	2.758718	6.755891	1.545712	0.482564
Std. dev	15.555957	0.890188	0.933409	1.181972	1.241663	3.346327	1.084641	0.499932
Variance	241.987795	0.792435	0.871252	1.397059	1.541728	11.197902	1.176446	0.249932

Test data

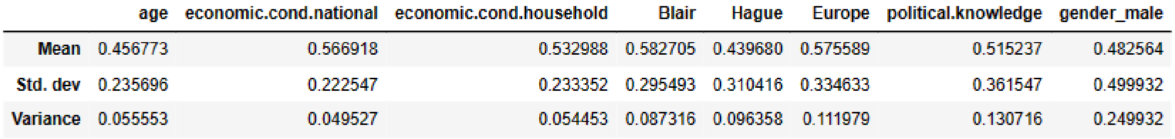


	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
Mean	54.460526	3.192982	3.151316	3.346491	2.728070	6.703947	1.528509	0.432018
Std. dev	16.050969	0.860638	0.926484	1.159058	1.211919	3.189624	1.084992	0.495901
Variance	257.633603	0.740698	0.858372	1.343416	1.468749	10.173699	1.177207	0.245918

Scaled data

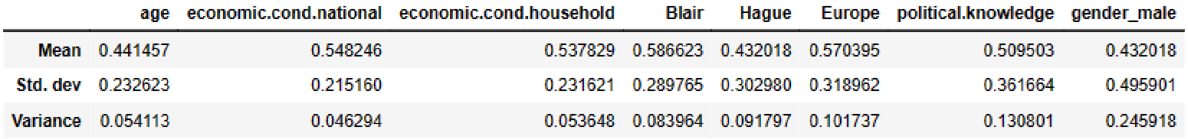
Table 16: Table showing standard deviation and variance of scaled data

Train data



	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
Mean	0.456773	0.566918	0.532988	0.582705	0.439680	0.575589	0.515237	0.482564
Std. dev	0.235696	0.222547	0.233352	0.295493	0.310416	0.334633	0.361547	0.499932
Variance	0.055553	0.049527	0.054453	0.087316	0.096358	0.111979	0.130716	0.249932

Test data



	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
Mean	0.441457	0.548246	0.537829	0.586623	0.432018	0.570395	0.509503	0.432018
Std. dev	0.232623	0.215160	0.231621	0.289765	0.302980	0.318962	0.361664	0.495901
Variance	0.054113	0.046294	0.053648	0.083964	0.091797	0.101737	0.130801	0.245918

Observations:

- We can note that, both original and scaled data has same standard deviation and variance for nominal variable (‘gender_male’) i.e., because of using Min Max method to scale the data.

- Before scaling, age column had a standard deviation of 15.56 (train), 16.05(test) and variance of 241.99 (train), 257.63 (test) which reduced to standard deviation of 0.24 (train), 0.23 (test) and variance of 0.06(train), 0.05 (test) after scaling.

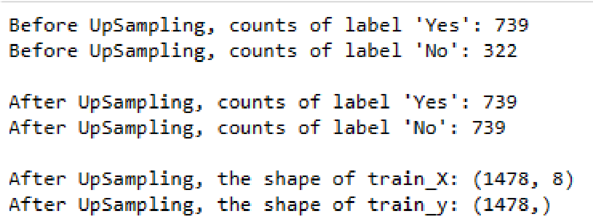
- In scaled data, all variables are in same measures, hence, improves accuracy and avoids wrong interpretation of the data.

1.4 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 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).

Ans:

- As the target class (vote column) have imbalanced data i.e., 1057: Labour and 460: Conservative, we can use SMOTE method in order to balance the data.
- We are using oversampling in order to balance the data, as under sampling will result in losing the data.
- Applying SMOTE method on the training data.

Table 17: Table showing samples before and after applying SMOTE (oversampling)

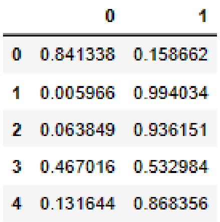


```
Before UpSampling, counts of label 'Yes': 739
Before UpSampling, counts of label 'No': 322

After UpSampling, counts of label 'Yes': 739
After UpSampling, counts of label 'No': 739

After UpSampling, the shape of train_X: (1478, 8)
After UpSampling, the shape of train_y: (1478,)
```

a) Logistic Regression:

- The data had 2 object datatype which is converted to categorical data.
- Applied Label Encoder to the target class, where ‘1’ represents ‘Labour’ party and ‘0’ represents Conservative party.
- Created dummy variable from already converted categorical data i.e., gender column.
- Assigned the independent variables to ‘X’ and dependent variables to ‘y’ and splitting the data into Train and Test at 70:30 ratio.
- Initializing and Fitting the **Logistic Regression Model**
- Predicting classes and Probability of the target variable

Table 18: Table showing predicted classes and Probability



	0	1
0	0.841338	0.158662
1	0.005966	0.994034
2	0.063849	0.936151
3	0.467016	0.532984
4	0.131644	0.868356

Model Evaluation:

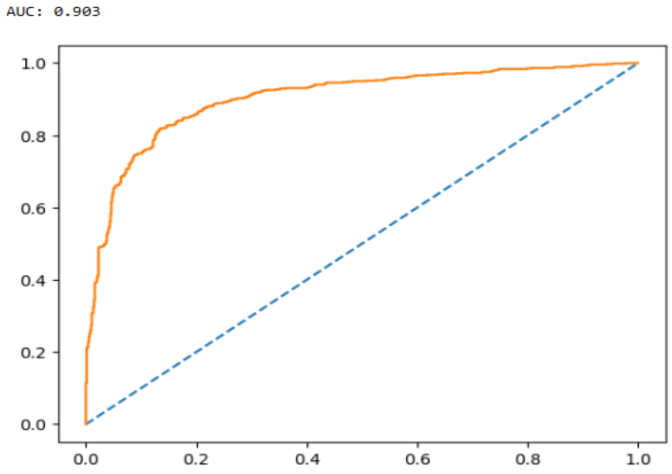
- **Accuracy:**

- Accuracy score of **Train** dataset is 0.83
- Accuracy score of **Test** dataset is 0.80

ROC Curve & ROC_AUC score:

- **Train** data:

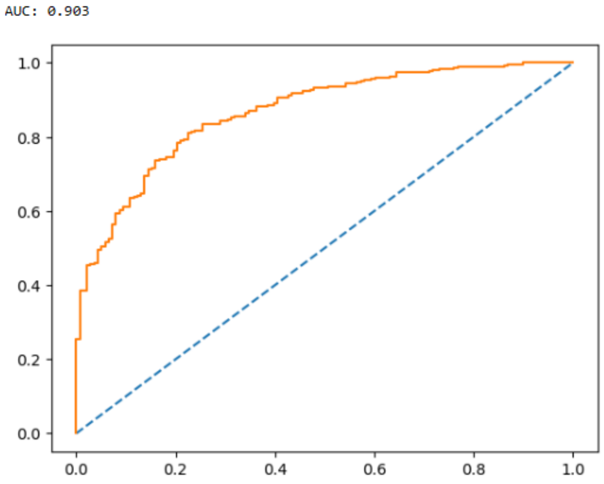
Figure 21: Figure showing ROC curve and ROC_AUC score for the Train data



- **ROC_AUC score for Train data is 0.903**

- **Test** data:

Figure 22: Figure showing ROC curve and ROC_AUC score for the Test data

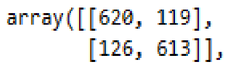


- **ROC_AUC score for Train data is 0.903**

<u>Confusion Matrix and Classification report:</u>

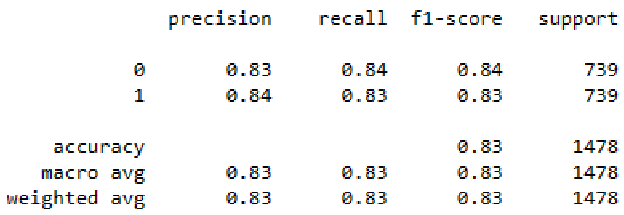
- **Train** data:

Table 19: Confusion matrix for the Train data of Logistic regression



```
array([[620, 119],
       [126, 613]],
```

Table 20: Classification report matrix for the Train data of Logistic regression



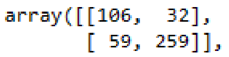
```
              precision    recall  f1-score   support

           0       0.83      0.84      0.84       739
           1       0.84      0.83      0.83       739

    accuracy                           0.83      1478
   macro avg       0.83      0.83      0.83      1478
weighted avg       0.83      0.83      0.83      1478
```

Observations:

- Model has correctly predicted 620 True positives and 613 True negatives and wrongly predicted 119 False negatives and 126 False positives.
- Accuracy score of **Train** dataset is 0.83.
- Precision: Model has predicted 84% of labour party votes and 83% of conservative party votes correctly.
- About 119 votes which were positive have been wrongly predicted as negatives. Similarly, 126 votes which are negative have been wrongly identified as positives.

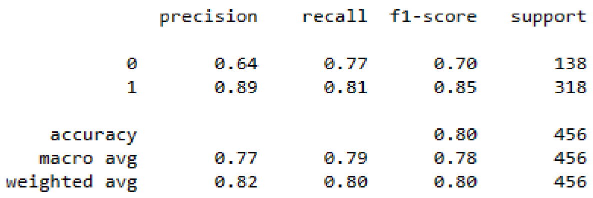
- **Test** data:

Table 21: Confusion matrix for the Test data of Logistic regression



```
array([[106,  32],
       [ 59, 259]],
```

Table 22: Classification report matrix for the Test data of Logistic regression



```
              precision    recall  f1-score   support

           0       0.64      0.77      0.70       138
           1       0.89      0.81      0.85       318

    accuracy                           0.80       456
   macro avg       0.77      0.79      0.78       456
weighted avg       0.82      0.80      0.80       456
```

Observations:

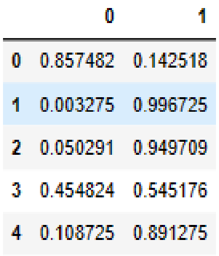
- 106 True positives and 259 True negatives have been predicted by the model.
- However, model have wrongly predicted 59 False positives and 32 False negatives.
- Accuracy score of **Train** dataset is 0.80.
- Precision: Model has predicted 89% of labour party votes and 64% of conservative party votes correctly.

- We can note that in test data conservative party has a precision of 64% i.e., model has failed to predict 36% of conservative party votes.

b) Linear Discriminant Analysis:

- The data had 2 object datatype which is converted to categorical data.
- Applied Label Encoder to the target class, where ‘1’ represents ‘Labour’ party and ‘0’ represents Conservative party.
- Created dummy variable from already converted categorical data i.e., gender column.
- Assigned the independent variables to ‘X’ and dependent variables to ‘y’ and splitting the data into Train and Test at 70:30 ratio.
- Initializing and Fitting the **LDA Model**
- Predicting classes and Probability of the target variable

Table 23: Table showing predicted classes and Probability



	0	1
0	0.857482	0.142518
1	0.003275	0.996725
2	0.050291	0.949709
3	0.454824	0.545176
4	0.108725	0.891275

Model Evaluation:

- **Accuracy:**

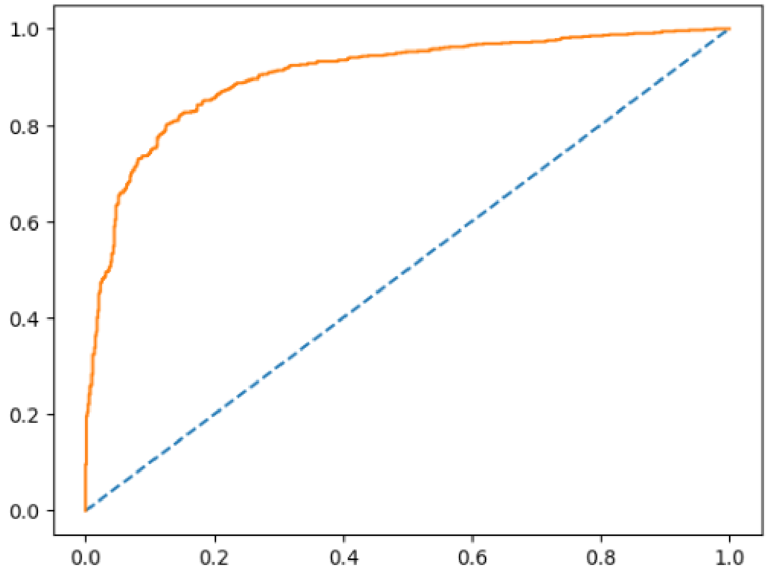
- Accuracy score of **Train** dataset is 0.83
- Accuracy score of **Test** dataset is 0.80

ROC Curve & ROC_AUC score:

- **Train** data:

Figure 23: Figure showing ROC curve and ROC_AUC score for the Train data

```
AUC: 0.903
```

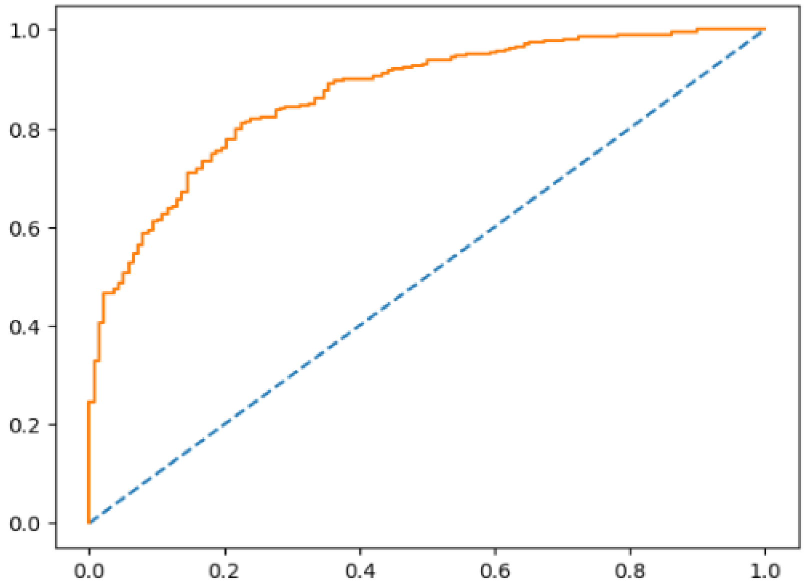


- **ROC_AUC score for Train data is 0.903**

- **Test** data:

Figure 24: Figure showing ROC curve and ROC_AUC score for the Test data

```
AUC: 0.903
```

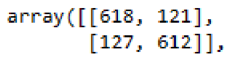


- **ROC_AUC score for Train data is 0.903**

Confusion Matrix and Classification report:

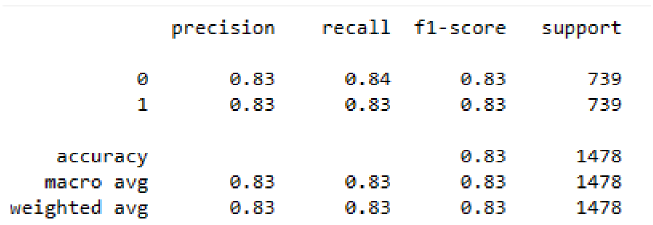
- **Train** data:

Table 24: Confusion matrix for the Train data of LDA model



```
array([[618, 121],
       [127, 612]],
```

Table 25: Classification report matrix for the Train data of LDA model



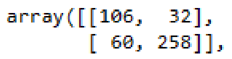
```
              precision    recall  f1-score   support

           0       0.83      0.84      0.83       739
           1       0.83      0.83      0.83       739

    accuracy                           0.83      1478
   macro avg       0.83      0.83      0.83      1478
weighted avg       0.83      0.83      0.83      1478
```

Observations:

- Model has correctly predicted 618 True positives and 612 True negatives and wrongly predicted 121 False negatives and 127 False positives.
- Accuracy score of **Train** dataset is 0.83.
- Precision: Model has correctly predicted 83% votes each of the labour and conservative party.
- About 121 votes which were positive have been wrongly predicted as negatives. Similarly, 127 votes which are negative have been wrongly identified as positives.

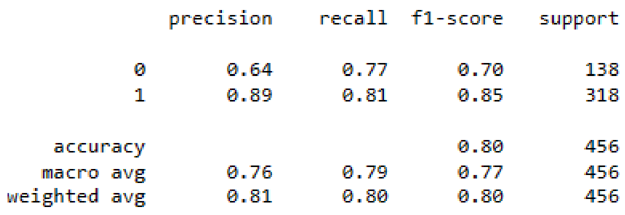
- **Test** data:

Table 26: Confusion matrix for the Test data of LDA model



```
array([[106,  32],
       [ 60, 258]],
```

Table 27: Classification report matrix for the Test data of LDA model



```
              precision    recall  f1-score   support

           0       0.64      0.77      0.70       138
           1       0.89      0.81      0.85       318

    accuracy                           0.80       456
   macro avg       0.76      0.79      0.77       456
weighted avg       0.81      0.80      0.80       456
```

Observations:

- 106 True positives and 258 True negatives have been predicted by the model.

- However, model have wrongly predicted 60 False positives and 32 False negatives.

- Accuracy score of **Train** dataset is 0.80.

- Precision: Model has predicted 89% of labour party votes and 64% of conservative party votes correctly.

- We can note that in test data, conservative party has a precision of 64% i.e., model has failed to predict 36% of conservative party votes. Even recall of the same is only 77% which was 84% in the train data.

Insights:

We may have to opt another model or need to improve the performance of the model by adopting various other means. For conservative party data it is overfitting for both the models. If we compare both these models, it has almost same precision, accuracy and recall rate, however, Logistic regression have slightly higher accuracy than the LDA model for the test data.

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).

Ans:

a) **<u>Gaussian Naïve Bayes:</u>**

- Naïve Bayes does not require the data to be scaled, hence we are using train and test data from the original data frame.
- After performing data preprocessing, we have split the data into train and test at 70:30 ratio.
- Initializing and Fitting the **Naïve Bayes Model.**
- Comparing Train and Test data.

- **<u>Model Evaluation:</u>**

  <u>Accuracy:</u>

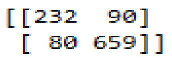
- Accuracy score of **Train** dataset is 0.84

- Accuracy score of **Test** dataset is 0.83

- **<u>Confusion Matrix and Classification report:</u>**

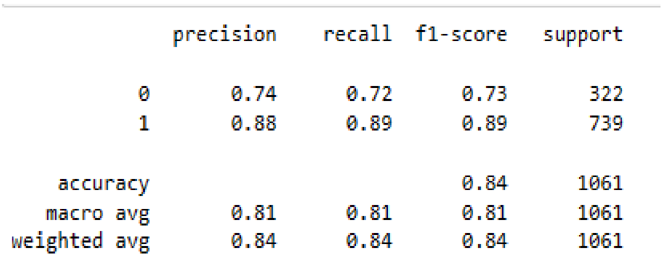
- **Train** data:

Table 28: Confusion matrix for the Train data of NB model



```
[[232  90]
 [ 80 659]]
```

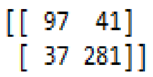
Table 29: Classification report matrix for the Train data of NB model



```
              precision    recall  f1-score   support

           0       0.74      0.72      0.73       322
           1       0.88      0.89      0.89       739

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

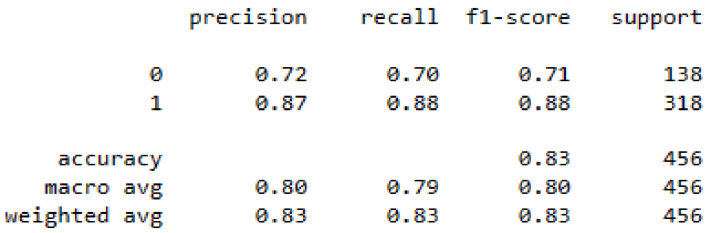
- **Test** data:

Table 30: Confusion matrix for the Test data of NB model



```
[[ 97  41]
 [ 37 281]]
```

Table 31: Classification report matrix for the Test data of NB model



```
              precision    recall  f1-score   support

           0       0.72      0.70      0.71       138
           1       0.87      0.88      0.88       318

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

Observations:

- Accuracy score is good for both train and test data.
- There is no overfitting issue as there is no much difference between the train and test data.
- Precision and recall rate of Labour party is good as the percentage reduced only by 1% from train to test data and the accuracy is 88% and 89% respectively.
- Precision and recall rate of Conservative party has to be improved as the accuracy of the prediction is around 72% and 70% in test data.
- We may able to improve the performance by balancing the data using SMOTE method.

b) KNN:

- KNN model requires the data to be scaled, hence we are using train and test data from the scaled data frame.
- After performing data preprocessing, we have split the data into train and test at 70:30 ratio.
- Initializing and Fitting the **KNN Model with k = 5 as default.**

- **Model Evaluation:**

  Accuracy:

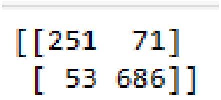
- Accuracy score of **Train** dataset when k=5 is 0.88
- Accuracy score of **Test** dataset when k=5 is 0.77

- **Confusion Matrix and Classification report:**

We are checking the performance of the train and test data when **k=5** (default)

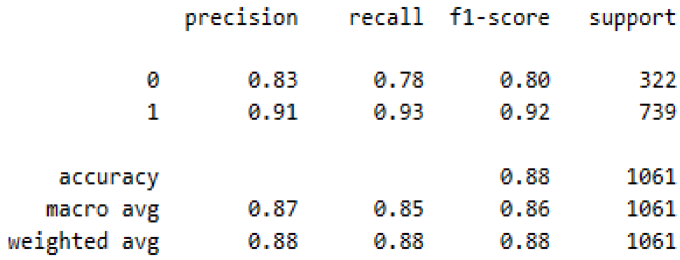
- **Train** data:

Table 32: Confusion matrix for the Train data (KNN model) k=5



```
[[251  71]
 [ 53 686]]
```

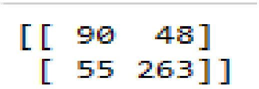
Table 33: Classification report matrix for the Train data (KNN model) k=5



```
              precision    recall  f1-score   support

           0       0.83      0.78      0.80       322
           1       0.91      0.93      0.92       739

    accuracy                           0.88      1061
   macro avg       0.87      0.85      0.86      1061
weighted avg       0.88      0.88      0.88      1061
```

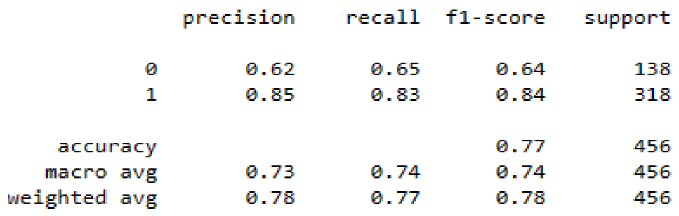
- **Test** data:

Table 34: Confusion matrix for the Test data (KNN model) k=5



```
[[ 90  48]
 [ 55 263]]
```

Table 35: Classification report matrix for the Test data (KNN model) k=5

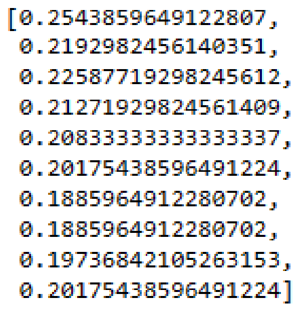


```
              precision    recall  f1-score   support

           0       0.62      0.65      0.64       138
           1       0.85      0.83      0.84       318

    accuracy                           0.77       456
   macro avg       0.73      0.74      0.74       456
weighted avg       0.78      0.77      0.78       456
```

- Accuracy score of **Train and Test** dataset when k=5 is 0.88 and 0.77 which is more than 10%, so we can improve the performance by changing the size of ‘k.’

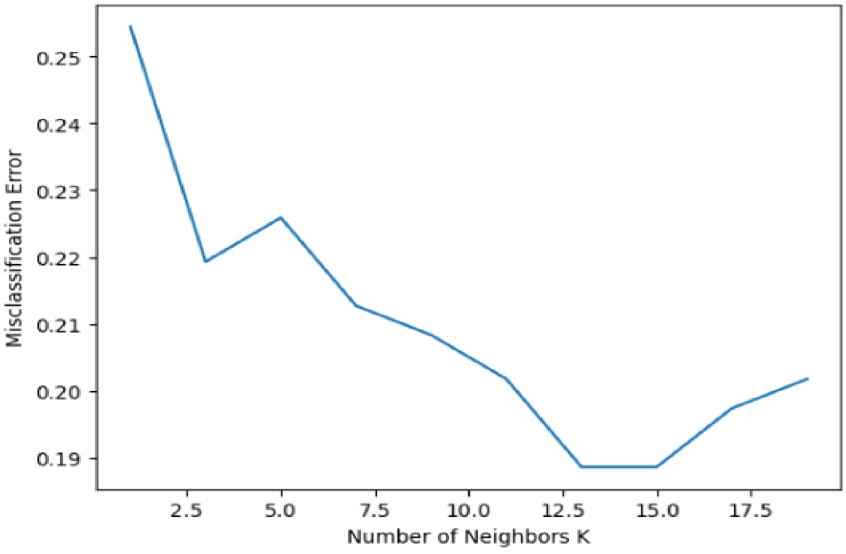
- We can find the optimal number of neighbours by using MCE (Misclassification Error), where we select a range of ‘k’ values to be tested and choose ‘k’ based on the least MCE.

Table 36: MCE for each model with neighbours k=1,3,5…19



```
[0.2543859649122807,
 0.2192982456140351,
 0.22587719298245612,
 0.21271929824561409,
 0.20833333333333337,
 0.20175438596491224,
 0.1885964912280702,
 0.1885964912280702,
 0.19736842105263153,
 0.20175438596491224]
```

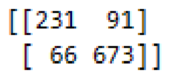
Figure 25: Plot showing MCE with neighbours k=1,3,5…19



- The plot shows the lowest MCE values at k=12 to k=15.
- We are opting k=15, based on the performance difference between Train and test data.
- **Confusion Matrix and Classification report:**

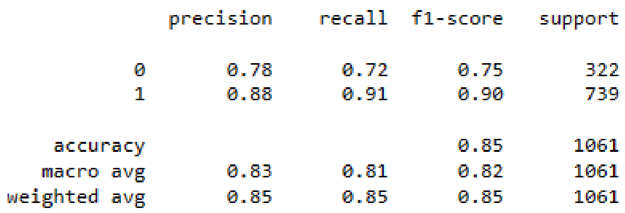
- **Train** data:

Table 37: Confusion matrix for the Train data (KNN model) k=15



```
[[231  91]
 [ 66 673]]
```

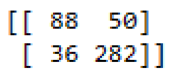
Table 38: Classification report matrix for the Train data (KNN model) k=15



```
              precision    recall  f1-score   support

           0       0.78      0.72      0.75       322
           1       0.88      0.91      0.90       739

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

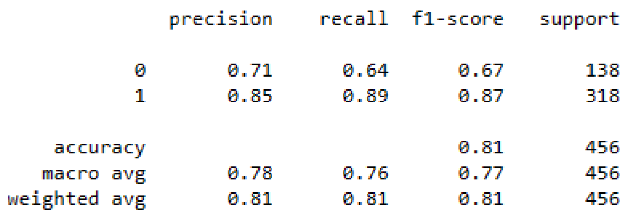
- **Test** data:

Table 39: Confusion matrix for the Test data (KNN model) k=15



```
[[ 88  50]
 [ 36 282]]
```

Table 40: Classification report matrix for the Test data (KNN model) k=15



```
              precision    recall  f1-score   support

           0       0.71      0.64      0.67       138
           1       0.85      0.89      0.87       318

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

Observations:

- Accuracy score of train and test data is 0.85 and 0.81.
- There is no overfitting issue as there is minimal difference between the train and test data.
- Precision rate has reduced to 71% (test) from 78% (train) for Conservative party data whereas precision is good for labour party as the train data has accuracy of 88% and test data has 85%.
- Recall rate is low for conservative party data as the accuracy of the prediction is 72% for train data and 64% for the test data.
- We may able to improve the performance by balancing the data using SMOTE method.

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.

Ans:

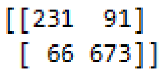
a) **KNN model:**

- **BASE model**

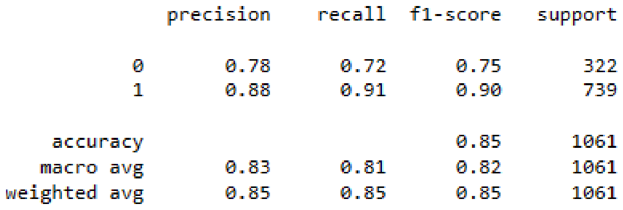
- Initializing and Fitting the **KNN Model with k = 15.**
- **Confusion Matrix and Classification report:**

- **Train** data:

Table 41: Confusion matrix and Classification report matrix for the Train data (KNN model) k=15



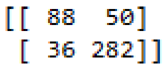
```
[[231  91]
 [ 66 673]]
```



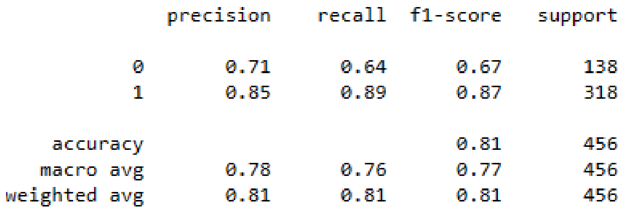
```
              precision    recall  f1-score   support

           0       0.78      0.72      0.75       322
           1       0.88      0.91      0.90       739

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

- **Test** data:

Table 42: Confusion matrix and Classification report matrix for the Test data (KNN model) k=15



```
[[ 88  50]
 [ 36 282]]
```



```
              precision    recall  f1-score   support

           0       0.71      0.64      0.67       138
           1       0.85      0.89      0.87       318

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

- **Accuracy score of train data is 85% and test data is 81%.**

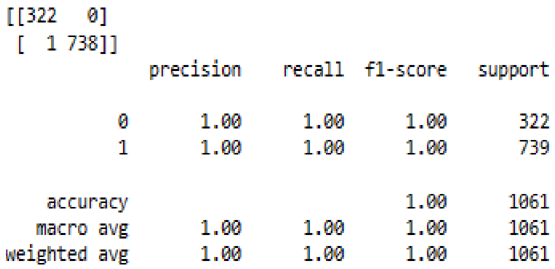
- ## GridsearchCV for KNN model

- In order to improve the performance of the model we are adding few additional parameters such as weights, n_neighbours and distance.
- Nearest neighbours considered are k=9, 11, 13, 15, 17.
- For distance matrix we have considered Euclidean and Manhattan.
- Initializing and fitting the **KNeighborsClassifier.**

- **Confusion Matrix and Classification report of GridsearchCV:**

- **Train** data:

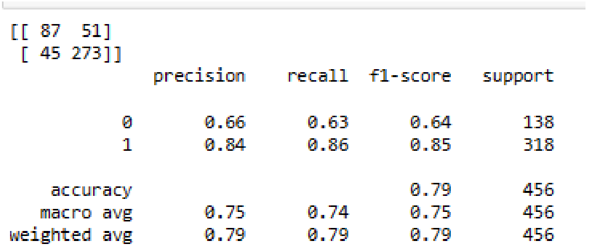
Table 43: Confusion matrix and Classification report matrix for the GridsearchCV Train data (KNN model)



```
[[322   0]
 [  1 738]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       322
           1       1.00      1.00      1.00       739

    accuracy                           1.00      1061
   macro avg       1.00      1.00      1.00      1061
weighted avg       1.00      1.00      1.00      1061
```

- **Test** data:

Table 44: Confusion matrix and Classification report matrix for the GridsearchCV Test data (KNN model)



```
[[ 87  51]
 [ 45 273]]
              precision    recall  f1-score   support

           0       0.66      0.63      0.64       138
           1       0.84      0.86      0.85       318

    accuracy                           0.79       456
   macro avg       0.75      0.74      0.75       456
weighted avg       0.79      0.79      0.79       456
```

- **Accuracy score of train data is 100% and test data is 79%.**

Observations:

- We can note that, GridsearchCV is highly overfitted as training data has 100% accuracy and test data it is reduced to 79%.

- In this case, we can go ahead with the base KNN model than gridsearchCV as the accuracy in base model is 85% (train) and 81% (test).
- Also, precision and recall are better in base model.

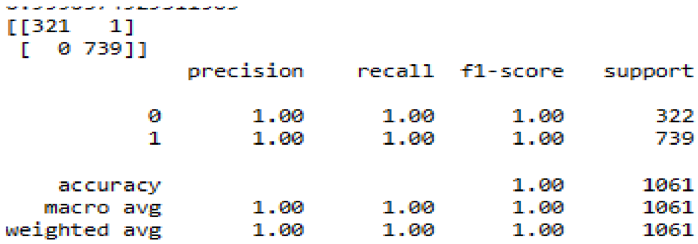
b) Random Forest classifier model:

- BASE model

- Initializing and Fitting Random Forest classifier.
- **Confusion Matrix and Classification report:**

- **Train** data:

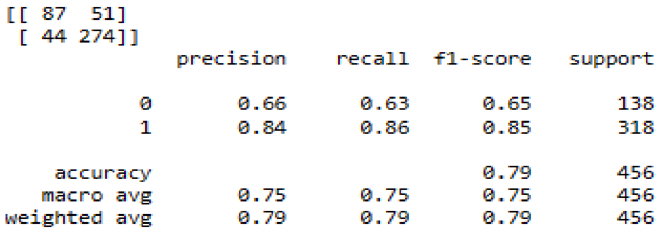
Table 45: Confusion matrix and Classification report matrix for the Train data Random Forest classifier



```
[[321   1]
 [  0 739]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       322
           1       1.00      1.00      1.00       739

    accuracy                           1.00      1061
   macro avg       1.00      1.00      1.00      1061
weighted avg       1.00      1.00      1.00      1061
```

- **Test** data:

Table 46: Confusion matrix and Classification report matrix for the Test data Random Forest classifier



```
[[ 87  51]
 [ 44 274]]
              precision    recall  f1-score   support

           0       0.66      0.63      0.65       138
           1       0.84      0.86      0.85       318

    accuracy                           0.79       456
   macro avg       0.75      0.75      0.75       456
weighted avg       0.79      0.79      0.79       456
```

- **Accuracy score of train data is 100% and test data is 79%.**

Observations:

- We can note that, base model has 100% accuracy for the train data and 79% for the test data.
- It is highly overfitted as accuracy of training is more than test i.e., around 21% difference.

GridsearchCV for Random Forest classifier

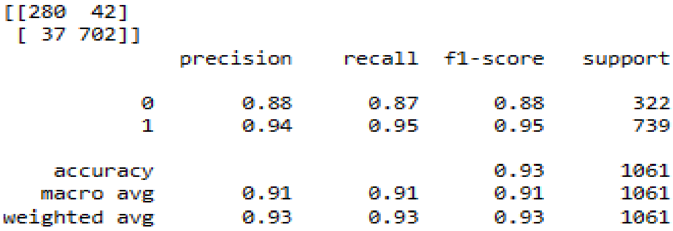
- In order to improve the performance of the model we are adding few additional parameters such as n_estimators, max features, max depth and criterion.

- Initializing and fitting the **RandomForestClassifier.**

- **Confusion Matrix and Classification report of GridsearchCV:**

- **Train** data:

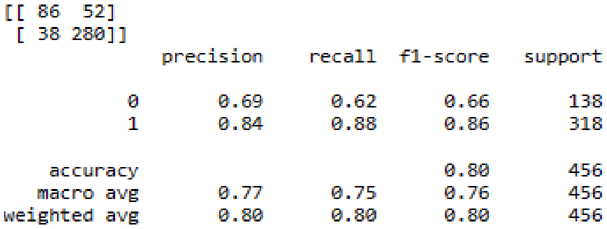
Table 47: Confusion matrix and Classification report matrix for the GridsearchCV Train data (Random Forest classifier)



```
[[280  42]
 [ 37 702]]
              precision    recall  f1-score   support

           0       0.88      0.87      0.88       322
           1       0.94      0.95      0.95       739

    accuracy                           0.93      1061
   macro avg       0.91      0.91      0.91      1061
weighted avg       0.93      0.93      0.93      1061
```

- **Test** data:

Table 48: Confusion matrix and Classification report matrix for the GridsearchCV Test data (Random Forest classifier)



```
[[ 86  52]
 [ 38 280]]
              precision    recall  f1-score   support

           0       0.69      0.62      0.66       138
           1       0.84      0.88      0.86       318

    accuracy                           0.80       456
   macro avg       0.77      0.75      0.76       456
weighted avg       0.80      0.80      0.80       456
```

- **Accuracy score of train data is 93% and test data is 80%.**

Observations:

- We can note that, GridsearchCV is overfitted as difference between accuracy of train and test data is around 13%.

- Precision and recall are also less for conservative data in GridsearchCV.

- However, GridsearchCV is better when we compare the same with the base model.

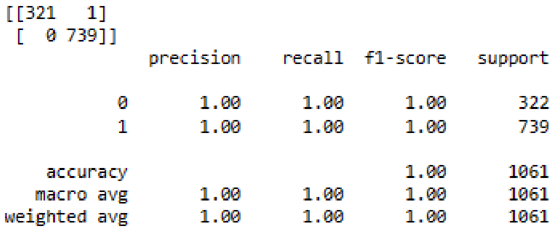
c) Bagging with Random Forest classifier:

- BASE model

- Initializing and Fitting Bagging classifier**.**
- **Confusion Matrix and Classification report:**

- **Train** data:

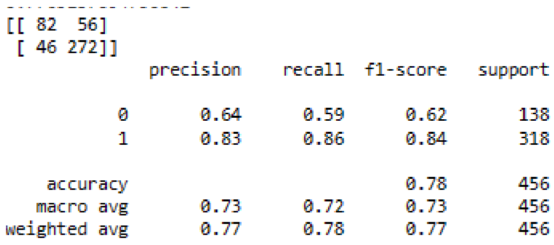
Table 49: Confusion matrix and Classification report matrix for the Train data Bagging classifier



```
[[321   1]
 [  0 739]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       322
           1       1.00      1.00      1.00       739

    accuracy                           1.00      1061
   macro avg       1.00      1.00      1.00      1061
weighted avg       1.00      1.00      1.00      1061
```

- **Test** data:

Table 50: Confusion matrix and Classification report matrix for the Test data Bagging classifier



```
[[ 82  56]
 [ 46 272]]
              precision    recall  f1-score   support

           0       0.64      0.59      0.62       138
           1       0.83      0.86      0.84       318

    accuracy                           0.78       456
   macro avg       0.73      0.72      0.73       456
weighted avg       0.77      0.78      0.77       456
```

- **Accuracy score of train data is 100% and test data is 78%.**

Observations:

- We can note that, base model has 100% accuracy for the train data and 78% for the test data.
- It is highly overfitted as accuracy of training is more than test i.e., around 22% difference.
- Precision and recall are also less in the test data.

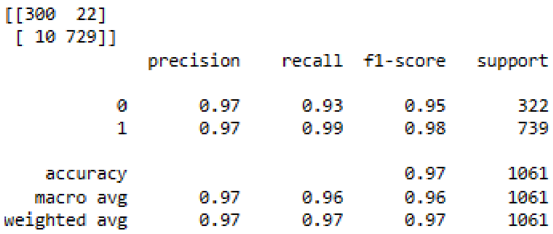
GridsearchCV for Bagging classifier

- In order to improve the performance of the model we are adding few additional parameters such as n_estimators and random state as 123.
- Initializing and fitting the **Bagging Classifier.**

- **Confusion Matrix and Classification report of GridsearchCV:**

- **Train** data:

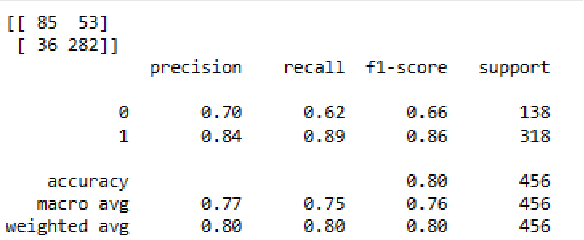
Table 51: Confusion matrix and Classification report matrix for the GridsearchCV Train data (Bagging classifier)



```
[[300  22]
 [ 10 729]]
              precision    recall  f1-score   support

           0       0.97      0.93      0.95       322
           1       0.97      0.99      0.98       739

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

- **Test** data:

Table 52: Confusion matrix and Classification report matrix for the GridsearchCV Test data (Bagging classifier)



```
[[ 85  53]
 [ 36 282]]
              precision    recall  f1-score   support

           0       0.70      0.62      0.66       138
           1       0.84      0.89      0.86       318

    accuracy                           0.80       456
   macro avg       0.77      0.75      0.76       456
weighted avg       0.80      0.80      0.80       456
```

- **Accuracy score of train data is 97% and test data is 80%.**

Observations:

- We can note that, GridsearchCV is overfitted as difference between accuracy of train and test data is around 17%.
- Precision and recall are also less for conservative data in GridsearchCV.
- However, GridsearchCV is better when we compare the same with the base model.

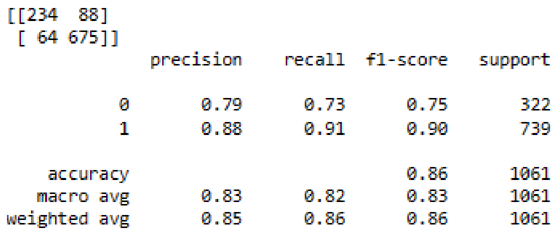
d) AdaBoosting:

- BASE model

- Initializing and Fitting AdaBoost classifier**.**
- **Confusion Matrix and Classification report:**

- **Train** data:

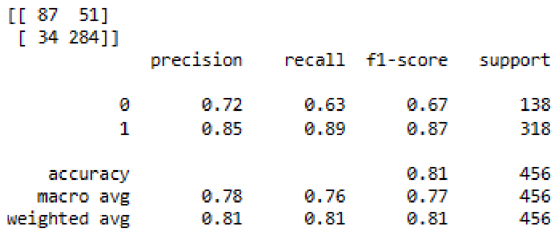
Table 53: Confusion matrix and Classification report matrix for the Train data AdaBoost classifier



```
[[234  88]
 [ 64 675]]
              precision    recall  f1-score   support

           0       0.79      0.73      0.75       322
           1       0.88      0.91      0.90       739

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

- **Test** data:

Table 54: Confusion matrix and Classification report matrix for the Test data AdaBoost classifier



```
[[ 87  51]
 [ 34 284]]
              precision    recall  f1-score   support

           0       0.72      0.63      0.67       138
           1       0.85      0.89      0.87       318

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

- **Accuracy score of train data is 86% and test data is 81%.**

Observations:

- Recall prediction percentage has to be improved on the test data.
- Also, need to reduce the False negatives and False positives.
- Accuracy of train and test is good i.e., train data has accuracy of 86% and test data has 81%.

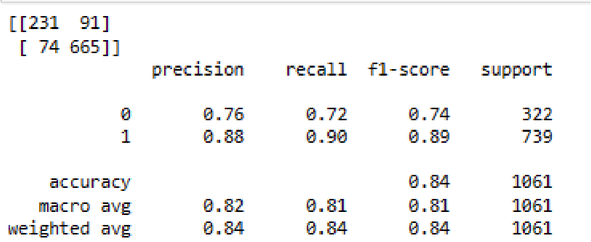
GridsearchCV for AdaBoost classifier

- In order to improve the performance of the model we are adding few additional parameters such as n_estimators, learning rate and have used SAMME algorithm on the model.
- Initializing and fitting the **AdaBoost Classifier.**

- **Confusion Matrix and Classification report of GridsearchCV:**

- **Train** data:

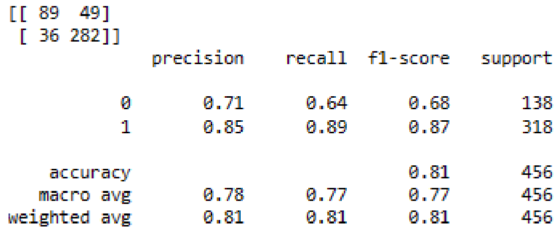
Table 55: Confusion matrix and Classification report matrix for the GridsearchCV Train data (AdaBoost classifier)



```
[[231  91]
 [ 74 665]]
              precision    recall  f1-score   support

           0       0.76      0.72      0.74       322
           1       0.88      0.90      0.89       739

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

- **Test** data:

Table 56: Confusion matrix and Classification report matrix for the GridsearchCV Test data (AdaBoost classifier)



```
[[ 89  49]
 [ 36 282]]
              precision    recall  f1-score   support

           0       0.71      0.64      0.68       138
           1       0.85      0.89      0.87       318

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

- **Accuracy score of train data is 84% and test data is 81%.**

Observations:

- Both the base model and gridsearchCV has good accuracy scores.
- However, both the models have to improve the performance and need to lessen the false negatives and false positives.
- Precision and recall are also less for conservative data in GridsearchCV.
- However, GridsearchCV is slightly better when we compare the same with the base model.

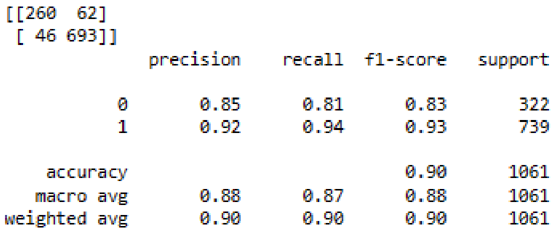
e) GradientBoost:

- BASE model

- Initializing and Fitting GardientBoost classifier**.**
- **Confusion Matrix and Classification report:**

- **Train** data:

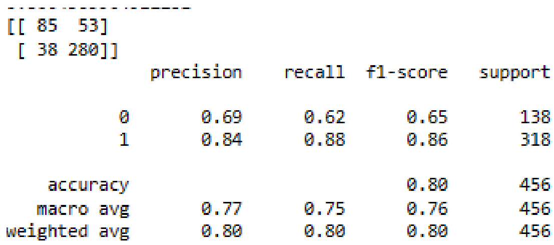
Table 57: Confusion matrix and Classification report matrix for the Train data GradientBoost classifier



```
[[260  62]
 [ 46 693]]
              precision    recall  f1-score   support

           0       0.85      0.81      0.83       322
           1       0.92      0.94      0.93       739

    accuracy                           0.90      1061
   macro avg       0.88      0.87      0.88      1061
weighted avg       0.90      0.90      0.90      1061
```

- **Test** data:

Table 58: Confusion matrix and Classification report matrix for the Test data GradientBoost classifier



```
------------------------
[[ 85  53]
 [ 38 280]]
              precision    recall  f1-score   support

           0       0.69      0.62      0.65       138
           1       0.84      0.88      0.86       318

    accuracy                           0.80       456
   macro avg       0.77      0.75      0.76       456
weighted avg       0.80      0.80      0.80       456
```

- **Accuracy score of train data is 90% and test data is 80%.**

Observations:

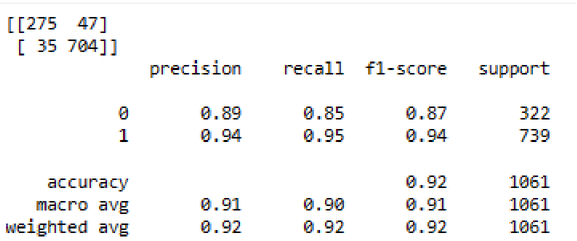
- Base model has accuracy of 90% on train data and 80% on test data which is a good score.
- Precision and recall of conservative data are less on both train and test data.
- Both False negatives and False positives has to be further reduced by adding additional parameters.

GridsearchCV for GradientBoost classifier

- In order to improve the performance of the model we are adding few additional parameters such as n_estimators and learning rate on the model.
- Initializing and fitting the **GradientBoost Classifier.**
- **Confusion Matrix and Classification report of GridsearchCV:**

- **Train** data:

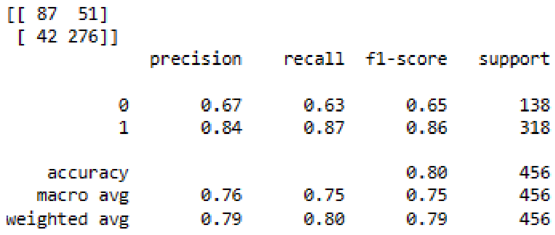
Table 59: Confusion matrix and Classification report matrix for the GridsearchCV Train data (GradientBoost classifier)



```
[[275  47]
 [ 35 704]]
              precision    recall  f1-score   support

           0       0.89      0.85      0.87       322
           1       0.94      0.95      0.94       739

    accuracy                           0.92      1061
   macro avg       0.91      0.90      0.91      1061
weighted avg       0.92      0.92      0.92      1061
```

- **Test** data:

Table 60: Confusion matrix and Classification report matrix for the GridsearchCV Test data (GradientBoost classifier)



```
[[ 87  51]
 [ 42 276]]
              precision    recall  f1-score   support

           0       0.67      0.63      0.65       138
           1       0.84      0.87      0.86       318

    accuracy                           0.80       456
   macro avg       0.76      0.75      0.75       456
weighted avg       0.79      0.80      0.79       456
```

- **Accuracy score of train data is 92% and test data is 80%.**

Observations:

- Both the base model and gridsearchCV has good accuracy scores.
- As the base model even gridsearchCV has issues with the false negatives and false positives.
- Both the models have almost same accuracy score on test data.
- However, base model is slightly better when we compare the same with the base model.

We can also additionally balance the data by using SMOTE method and proceed with the above models to have a slight improvement on the performance of the models.

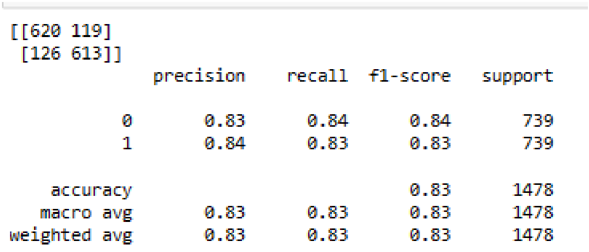
f) Logistic Regression:

- BASE model

- Initializing and Fitting Logistic Regression classifier**.**
- **Confusion Matrix and Classification report:**

- **Train** data:

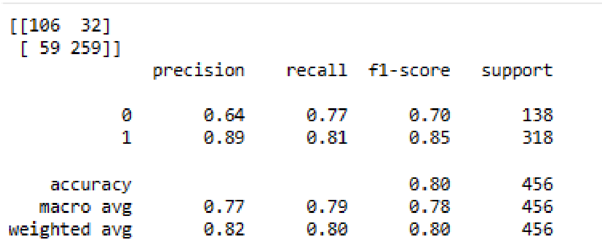
Table 61: Confusion matrix and Classification report matrix for the Train data LogisticRegression classifier



```
[[620 119]
 [126 613]]
              precision    recall  f1-score   support

           0       0.83      0.84      0.84       739
           1       0.84      0.83      0.83       739

    accuracy                           0.83      1478
   macro avg       0.83      0.83      0.83      1478
weighted avg       0.83      0.83      0.83      1478
```

- **Test** data:

Table 62: Confusion matrix and Classification report matrix for the Test data LogisticRegression classifier



```
[[106  32]
 [ 59 259]]
              precision    recall  f1-score   support

           0       0.64      0.77      0.70       138
           1       0.89      0.81      0.85       318

    accuracy                           0.80       456
   macro avg       0.77      0.79      0.78       456
weighted avg       0.82      0.80      0.80       456
```

- **Accuracy score of train data is 83% and test data is 80%.**

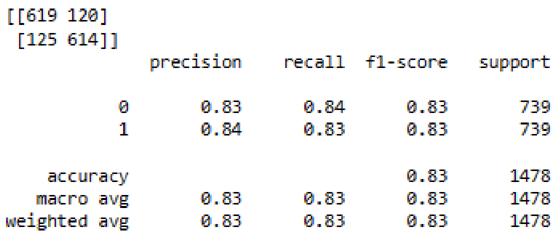
GridsearchCV for LogisticRegression classifier

- In order to improve the performance of the model we are adding few additional parameters such as ‘penalty’, ‘solver’ and ‘tol’ on the model.
- Initializing and fitting the **LogisticRegression Classifier.**

- <u>**Confusion Matrix and Classification report of GridsearchCV:**</u>

- **Train** data:

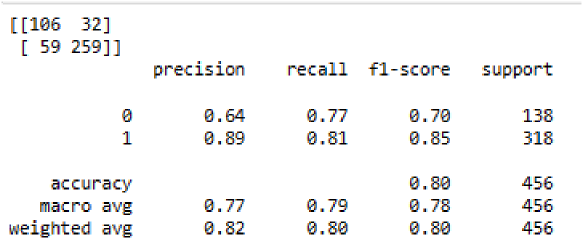
Table 63: Confusion matrix and Classification report matrix for the GridsearchCV Train data (LogisticRegression classifier)



```
[[619 120]
 [125 614]]
              precision    recall  f1-score   support

           0       0.83      0.84      0.83       739
           1       0.84      0.83      0.83       739

    accuracy                           0.83      1478
   macro avg       0.83      0.83      0.83      1478
weighted avg       0.83      0.83      0.83      1478
```

- **Test** data:

Table 64: Confusion matrix and Classification report matrix for the GridsearchCV Test data (LogisticRegression classifier)



```
[[106  32]
 [ 59 259]]
              precision    recall  f1-score   support

           0       0.64      0.77      0.70       138
           1       0.89      0.81      0.85       318

    accuracy                           0.80       456
   macro avg       0.77      0.79      0.78       456
weighted avg       0.82      0.80      0.80       456
```

- **Accuracy score of train data is 83% and test data is 80%.**

Observations:

- We have balanced the data by using SMOTE upsampling method.
- Both the base model and gridsearchCV has good accuracy scores.
- Both the models have almost same accuracy score on test data.
- Precision of conservative party data has to be improved by using other parameters.

g) <u>LDA:</u>

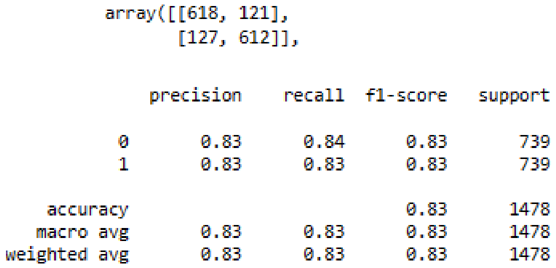
- <u>**BASE model**</u>

- Initializing and Fitting LDA classifier**.**

- **<u>Confusion Matrix and Classification report:</u>**

- **Train** data:

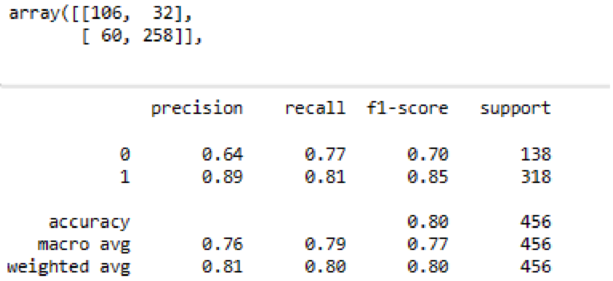
Table 65: Confusion matrix and Classification report matrix for the Train data LDA classifier



```
          array([[618, 121],
                 [127, 612]],

                  precision    recall  f1-score   support

           0       0.83      0.84      0.83       739
           1       0.83      0.83      0.83       739

    accuracy                           0.83      1478
   macro avg       0.83      0.83      0.83      1478
weighted avg       0.83      0.83      0.83      1478
```

- **Test** data:

Table 66: Confusion matrix and Classification report matrix for the Test data LDA classifier



```
array([[106,  32],
       [ 60, 258]],

              precision    recall  f1-score   support

           0       0.64      0.77      0.70       138
           1       0.89      0.81      0.85       318

    accuracy                           0.80       456
   macro avg       0.76      0.79      0.77       456
weighted avg       0.81      0.80      0.80       456
```

- **Accuracy score of train data is 83% and test data is 80%.**

<u>GridsearchCV for LDA classifier</u>

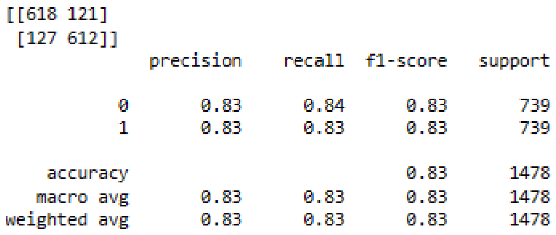
- In order to improve the performance of the model we are adding few additional parameters such as ‘solver’ and ‘tol’ on the model.

- Initializing and fitting the **LDA Classifier.**

- **<u>Confusion Matrix and Classification report of GridsearchCV:</u>**

- **Train** data:

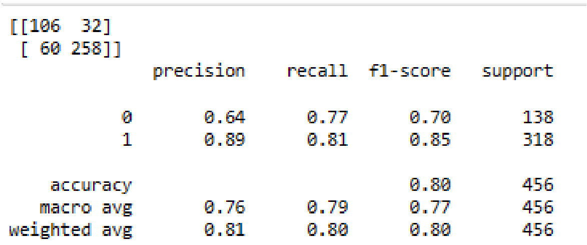
Table 67: Confusion matrix and Classification report matrix for the GridsearchCV Train data (LDA classifier)



```
[[618 121]
 [127 612]]
              precision    recall  f1-score   support

           0       0.83      0.84      0.83       739
           1       0.83      0.83      0.83       739

    accuracy                           0.83      1478
   macro avg       0.83      0.83      0.83      1478
weighted avg       0.83      0.83      0.83      1478
```

- **Test** data:

Table 68: Confusion matrix and Classification report matrix for the GridsearchCV Test data (LDA classifier)



```
[[106  32]
 [ 60 258]]
              precision    recall  f1-score   support

           0       0.64      0.77      0.70       138
           1       0.89      0.81      0.85       318

    accuracy                           0.80       456
   macro avg       0.76      0.79      0.77       456
weighted avg       0.81      0.80      0.80       456
```

- **Accuracy score of train data is 83% and test data is 80%.**

Observations:

- We have balanced the data by using SMOTE upsampling method.
- Both the models have same accuracy score on test data.
- Precision of conservative party data has to be improved by using other parameters.

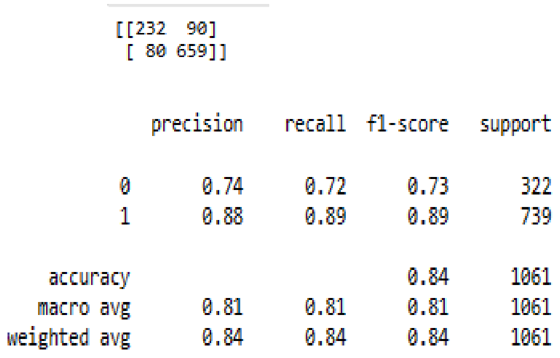
h) Naïve Bayes:

- BASE model

- Initializing and Fitting GaussianNB classifier**.**
- **Confusion Matrix and Classification report:**

- **Train** data:

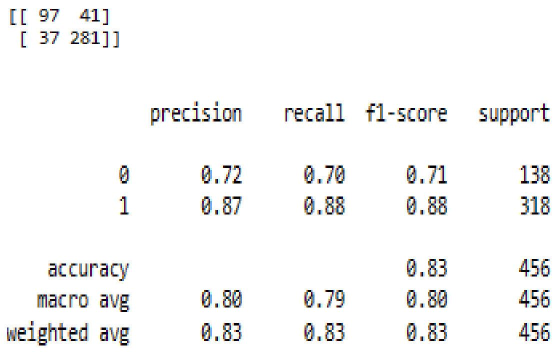
Table 69: Confusion matrix and Classification report matrix for the Train data GaussianNB classifier



```
[[232  90]
 [ 80 659]]

              precision    recall  f1-score   support

           0       0.74      0.72      0.73       322
           1       0.88      0.89      0.89       739

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

- **Test** data:

Table 70: Confusion matrix and Classification report matrix for the Test data GaussianNB classifier



```
[[ 97  41]
 [ 37 281]]

              precision    recall  f1-score   support

           0       0.72      0.70      0.71       138
           1       0.87      0.88      0.88       318

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

- **Accuracy score of train data is 84% and test data is 83%.**

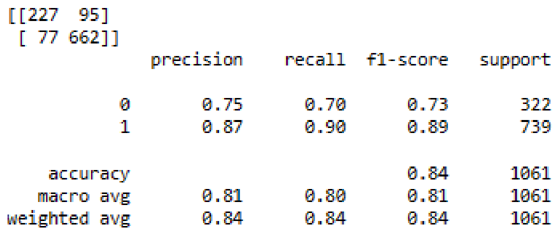
GridsearchCV for GaussianNB classifier

- In order to improve the performance of the model we are adding additional parameter such as ‘var_smoothing’ on the model.
- Initializing and fitting the **GaussianNB Classifier.**

- **Confusion Matrix and Classification report of GridsearchCV:**

- **Train** data:

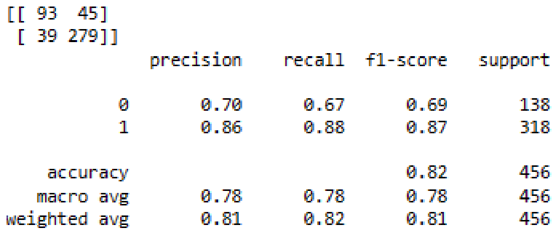
Table 71: Confusion matrix and Classification report matrix for the GridsearchCV Train data (GaussianNB classifier)



```
[[227  95]
 [ 77 662]]
              precision    recall  f1-score   support

           0       0.75      0.70      0.73       322
           1       0.87      0.90      0.89       739

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

- **Test** data:

Table 72: Confusion matrix and Classification report matrix for the GridsearchCV Test data (GaussianNB classifier)



```
[[ 93  45]
 [ 39 279]]
              precision    recall  f1-score   support

           0       0.70      0.67      0.69       138
           1       0.86      0.88      0.87       318

    accuracy                           0.82       456
   macro avg       0.78      0.78      0.78       456
weighted avg       0.81      0.82      0.81       456
```

- **Accuracy score of train data is 84% and test data is 82%.**

Observations:

- Both the models have almost same accuracy score on test data.
- GridsearchCV has low recall rate on the conservative party data.
- Base model has a good fit between the train and test data.

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).

Ans:

a) **KNN model:**

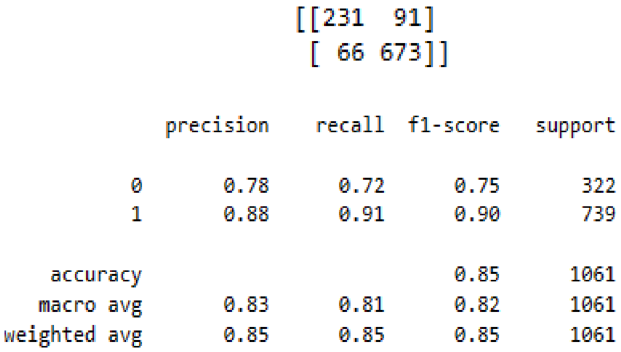
- **BASE model**

- Initializing and Fitting the **KNN Model.**
- **Accuracy**:
    - Accuracy score of **Train** dataset is 0.85
    - Accuracy score of **Test** dataset is 0.81

- **Confusion Matrix and Classification report:**

- **Train** data:

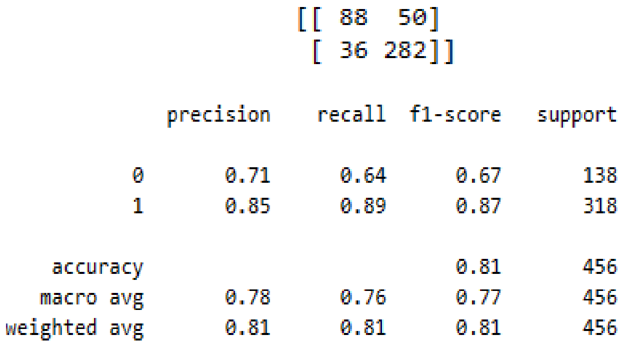
Table 73: Confusion matrix and Classification report matrix for the Train data (KNN model)



```
                      [[231  91]
                       [ 66 673]]

              precision    recall  f1-score   support

           0       0.78      0.72      0.75       322
           1       0.88      0.91      0.90       739

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

- **Test** data:

Table 74: Confusion matrix and Classification report matrix for the Test data (KNN model)

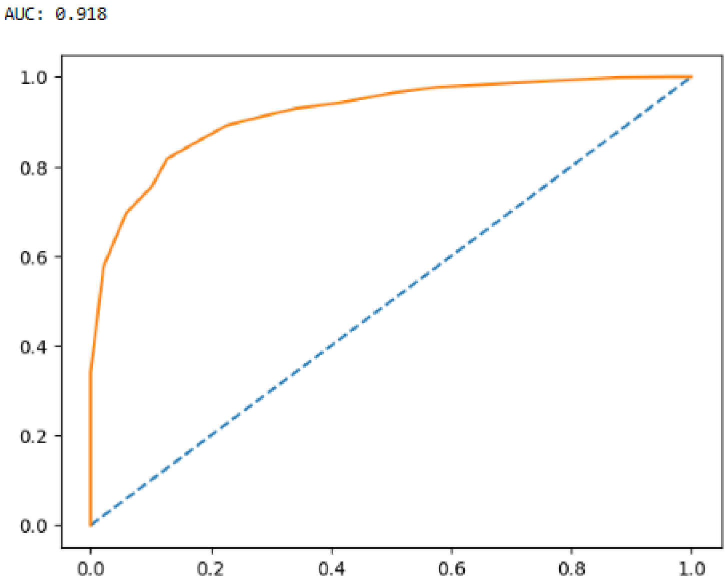


```
                      [[ 88  50]
                       [ 36 282]]

              precision    recall  f1-score   support

           0       0.71      0.64      0.67       138
           1       0.85      0.89      0.87       318

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

- **<u>ROC Curve & ROC_AUC score:</u>**

- **Train** data:

Figure 26: Figure showing ROC curve and ROC_AUC score for the KNN Train data

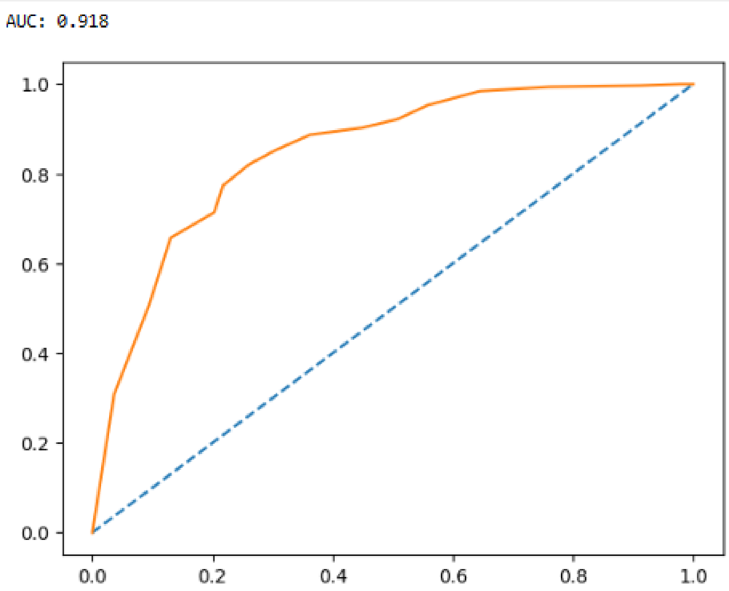


```
AUC: 0.918
```

- **ROC_AUC score for Train data is 0.918**

- **Test** data:

Figure 27: Figure showing ROC curve and ROC_AUC score for the KNN Test data



```
AUC: 0.918
```

- **ROC_AUC score for Test data is 0.918**

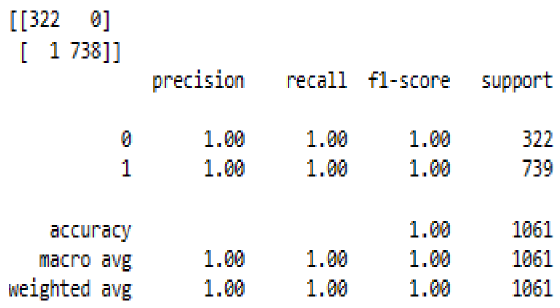
GridsearchCV for KNN model

- Initializing and fitting the **KNeighborsClassifier.**
- **Accuracy:**
  - Accuracy score of **Train** dataset is 100
  - Accuracy score of **Test** dataset is 0.79

- **Confusion Matrix and Classification report:**

- **Train** data:

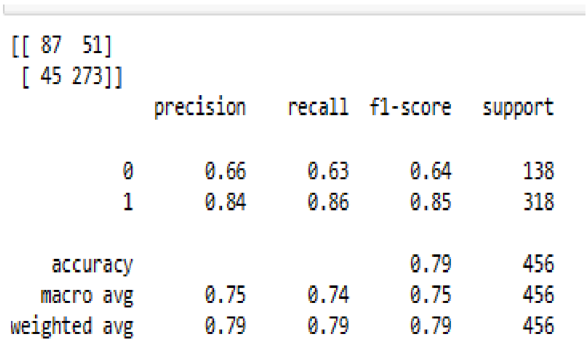
Table 75: Confusion matrix and Classification report matrix for the Train data (GridsearchCV KNN model)



```
[[322   0]
 [  1 738]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       322
           1       1.00      1.00      1.00       739

    accuracy                           1.00      1061
   macro avg       1.00      1.00      1.00      1061
weighted avg       1.00      1.00      1.00      1061
```

- **Test** data:

Table 76: Confusion matrix and Classification report matrix for the Test data (GridsearchCV KNN model)

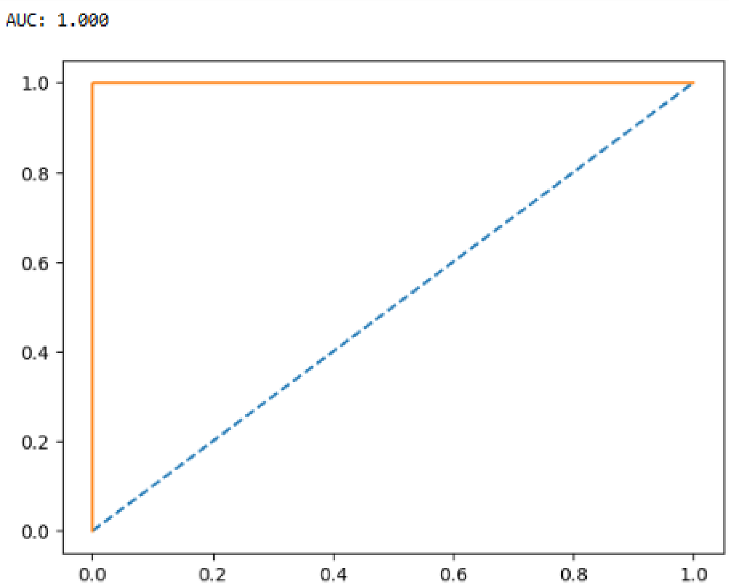


```
[[ 87  51]
 [ 45 273]]
              precision    recall  f1-score   support

           0       0.66      0.63      0.64       138
           1       0.84      0.86      0.85       318

    accuracy                           0.79       456
   macro avg       0.75      0.74      0.75       456
weighted avg       0.79      0.79      0.79       456
```

- **<u>ROC Curve & ROC_AUC score GridsearchCV:</u>**

- **Train** data:

Figure 28: Figure showing ROC curve and ROC_AUC score for the GridsearchCV KNN Train data

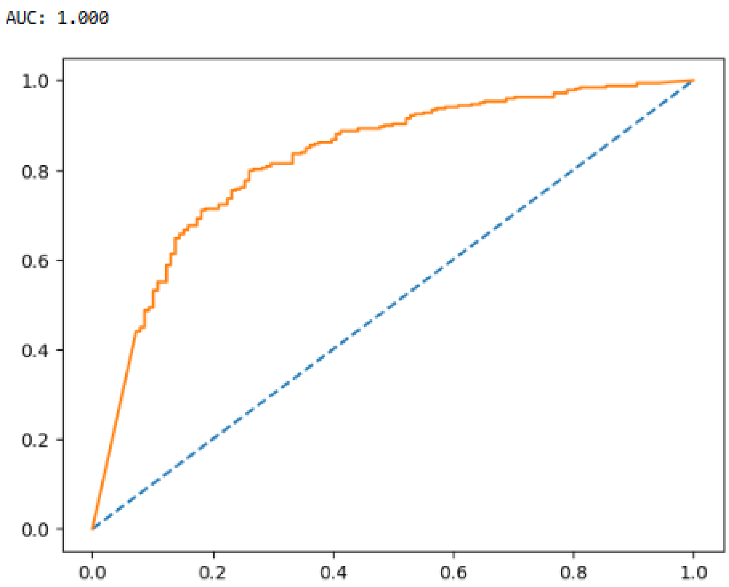


```
AUC: 1.000
```

- **ROC_AUC score for Train data is 1.00**

- **Test** data:

Figure 29: Figure showing ROC curve and ROC_AUC score for the GridsearchCV KNN Test data



```
AUC: 1.000
```

- **ROC_AUC score for Test data is 1.00**

b) <u>Random Forest classifier model:</u>

- <u>BASE model</u>

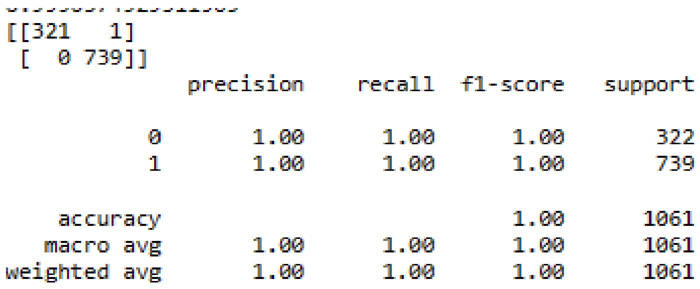
- Initializing and Fitting Random Forest classifier

  - **Accuracy:**
    - Accuracy score of **Train** dataset is 100
    - Accuracy score of **Test** dataset is 0.79

- **<u>Confusion Matrix and Classification report:</u>**

- **Train** data:

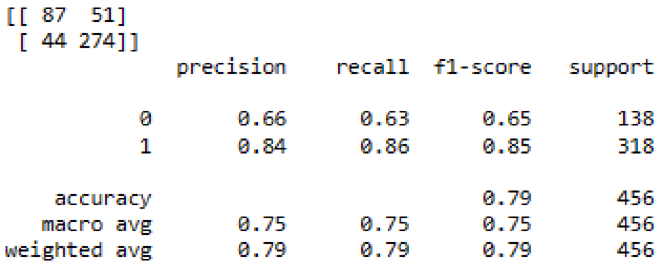
Table 77: Confusion matrix and Classification report matrix for the Train data (Random Forest)



```
[[321   1]
 [  0 739]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       322
           1       1.00      1.00      1.00       739

    accuracy                           1.00      1061
   macro avg       1.00      1.00      1.00      1061
weighted avg       1.00      1.00      1.00      1061
```

- **Test** data:

Table 78: Confusion matrix and Classification report matrix for the Test data (Random Forest)

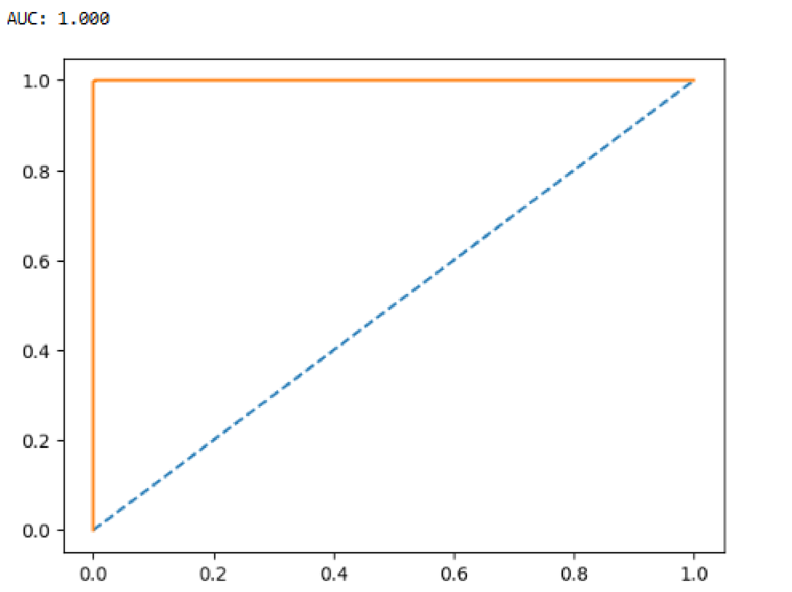


```
[[ 87  51]
 [ 44 274]]
              precision    recall  f1-score   support

           0       0.66      0.63      0.65       138
           1       0.84      0.86      0.85       318

    accuracy                           0.79       456
   macro avg       0.75      0.75      0.75       456
weighted avg       0.79      0.79      0.79       456
```

- **<u>ROC Curve & ROC_AUC score:</u>**

- **Train** data:

Figure 30: Figure showing ROC curve and ROC_AUC score for the Random Forest Train data

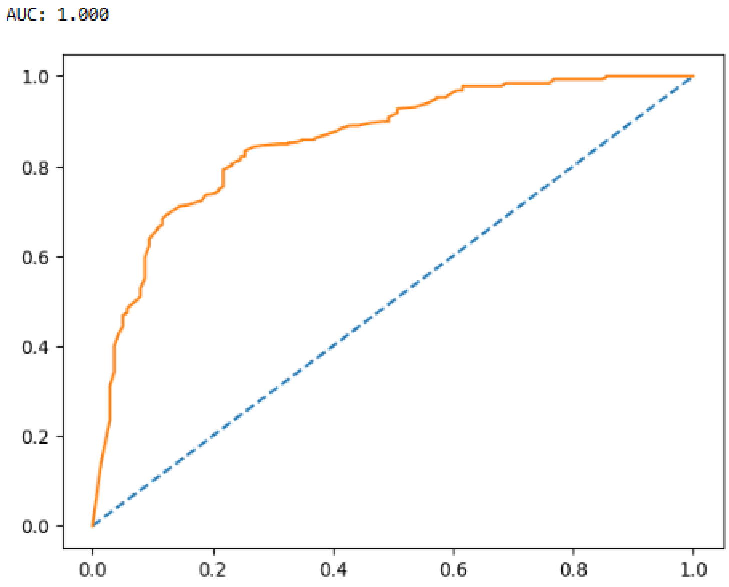


```
AUC: 1.000
```

- **ROC_AUC score for Train data is 1.00**

- **Test** data:

Figure 31: Figure showing ROC curve and ROC_AUC score for the Random Forest Test data



```
AUC: 1.000
```

- **ROC_AUC score for Test data is 1.00**

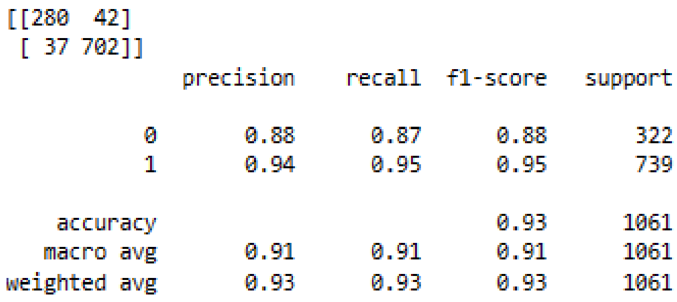
GridsearchCV for RandomForest model

- Initializing and fitting the **RandomForestClassifier.**
- **Accuracy:**
    - Accuracy score of **Train** dataset is 0.93
    - Accuracy score of **Test** dataset is 0.80

- **Confusion Matrix and Classification report:**

- **Train** data:

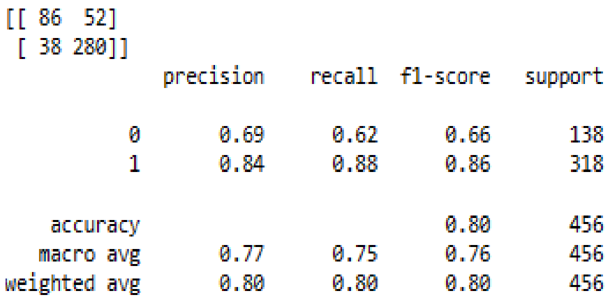
Table 79: Confusion matrix and Classification report matrix for the Train data (GridsearchCV Random Forest model)



```
[[280  42]
 [ 37 702]]
              precision    recall  f1-score   support

           0       0.88      0.87      0.88       322
           1       0.94      0.95      0.95       739

    accuracy                           0.93      1061
   macro avg       0.91      0.91      0.91      1061
weighted avg       0.93      0.93      0.93      1061
```

- **Test** data:

Table 80: Confusion matrix and Classification report matrix for the Test data (GridsearchCV Random Forest model)



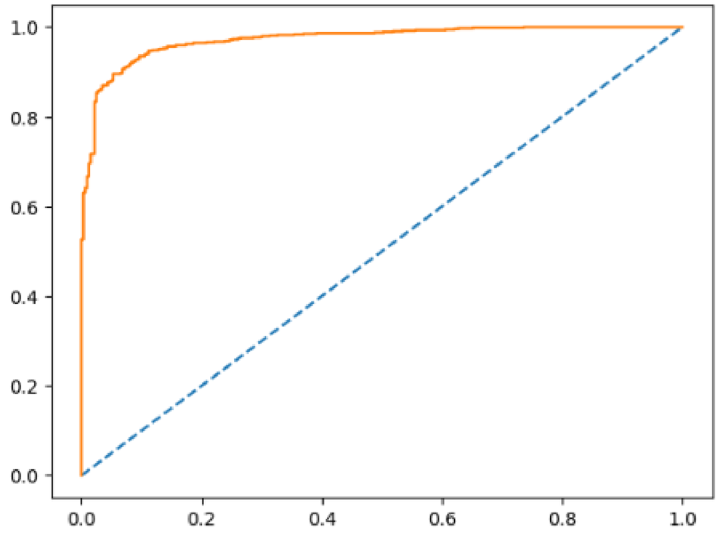
```
[[ 86  52]
 [ 38 280]]
              precision    recall  f1-score   support

           0       0.69      0.62      0.66       138
           1       0.84      0.88      0.86       318

    accuracy                           0.80       456
   macro avg       0.77      0.75      0.76       456
weighted avg       0.80      0.80      0.80       456
```

- **<u>ROC Curve & ROC_AUC score GridsearchCV:</u>**

- **Train** data:

Figure 32: Figure showing ROC curve and ROC_AUC score for the GridsearchCV Random Forest Train data

```
AUC: 0.973
```

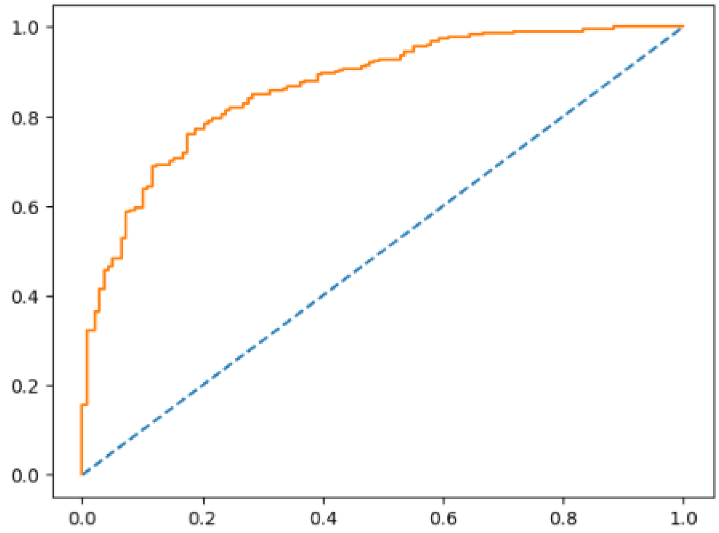


- **ROC_AUC score for Train data is 0.973**

- **Test** data:

Figure 33: Figure showing ROC curve and ROC_AUC score for the GridsearchCV Random Forest Test data

```
AUC: 0.973
```



- **ROC_AUC score for Test data is 0.973**

c) Bagging with Random Forest classifier:

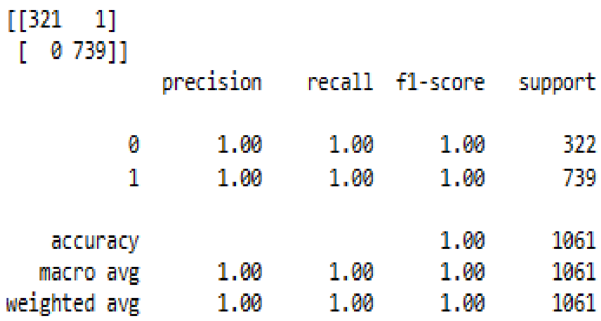
- BASE model

- Initializing and Fitting Bagging classifier**.**
  - **Accuracy:**
    - Accuracy score of **Train** dataset is 100
    - Accuracy score of **Test** dataset is 0.78

- **Confusion Matrix and Classification report:**

- **Train** data:

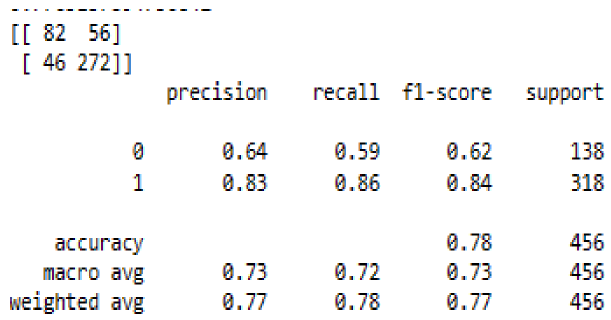
Table 81: Confusion matrix and Classification report matrix for the Train data (Bagging)



```
[[321   1]
 [  0 739]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       322
           1       1.00      1.00      1.00       739

    accuracy                           1.00      1061
   macro avg       1.00      1.00      1.00      1061
weighted avg       1.00      1.00      1.00      1061
```

- **Test** data:

Table 82: Confusion matrix and Classification report matrix for the Test data (Bagging)

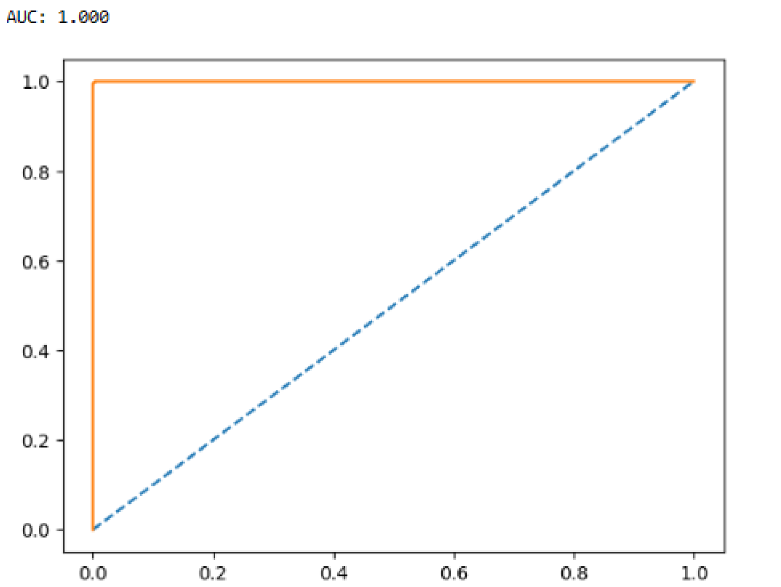


```
[[ 82  56]
 [ 46 272]]
              precision    recall  f1-score   support

           0       0.64      0.59      0.62       138
           1       0.83      0.86      0.84       318

    accuracy                           0.78       456
   macro avg       0.73      0.72      0.73       456
weighted avg       0.77      0.78      0.77       456
```

- <u>**ROC Curve & ROC_AUC score:**</u>

- **Train** data:

Figure 34: Figure showing ROC curve and ROC_AUC score for the Bagging Train data

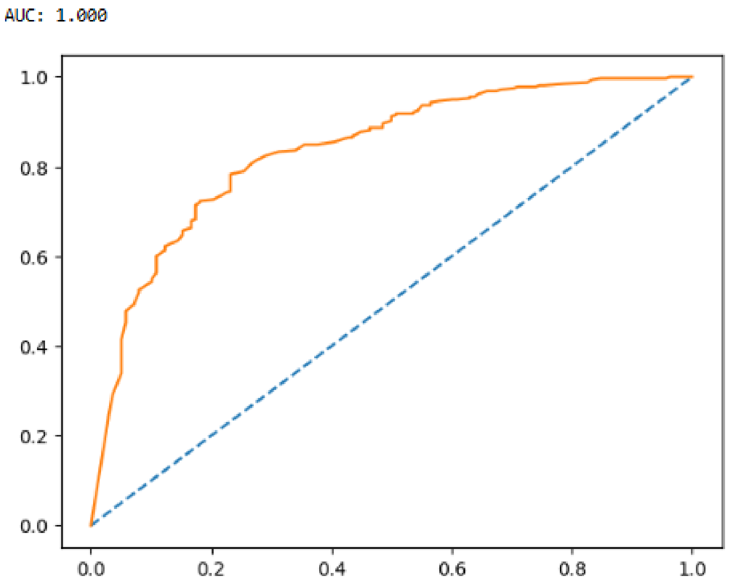


```
AUC: 1.000
```

- **ROC_AUC score for Train data is 1.00**

- **Test** data:

Figure 35: Figure showing ROC curve and ROC_AUC score for the Bagging Test data



```
AUC: 1.000
```

- **ROC_AUC score for Test data is 1.00**

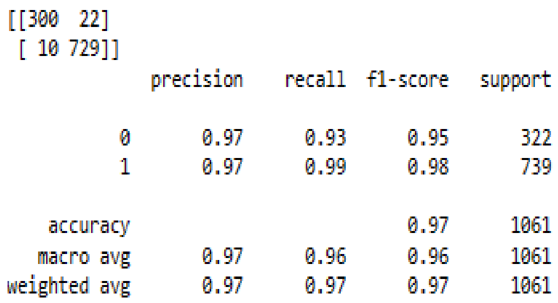
GridsearchCV for Bagging model

- Initializing and fitting the **BaggingClassifier.**
- **Accuracy:**
  - Accuracy score of **Train** dataset is 0.97
  - Accuracy score of **Test** dataset is 0.80

- **Confusion Matrix and Classification report:**

- **Train** data:

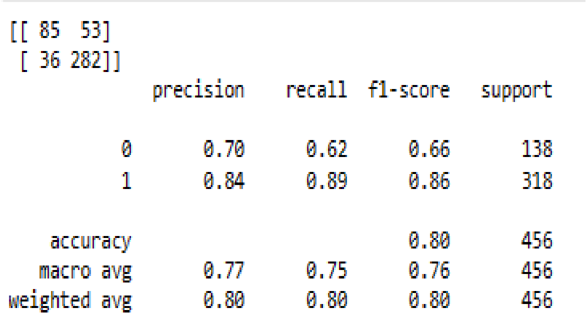
Table 83: Confusion matrix and Classification report matrix for the Train data (GridsearchCV Bagging model)



```
[[300  22]
 [ 10 729]]
              precision    recall  f1-score   support

           0       0.97      0.93      0.95       322
           1       0.97      0.99      0.98       739

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

- **Test** data:

Table 84: Confusion matrix and Classification report matrix for the Test data (GridsearchCV Bagging model)

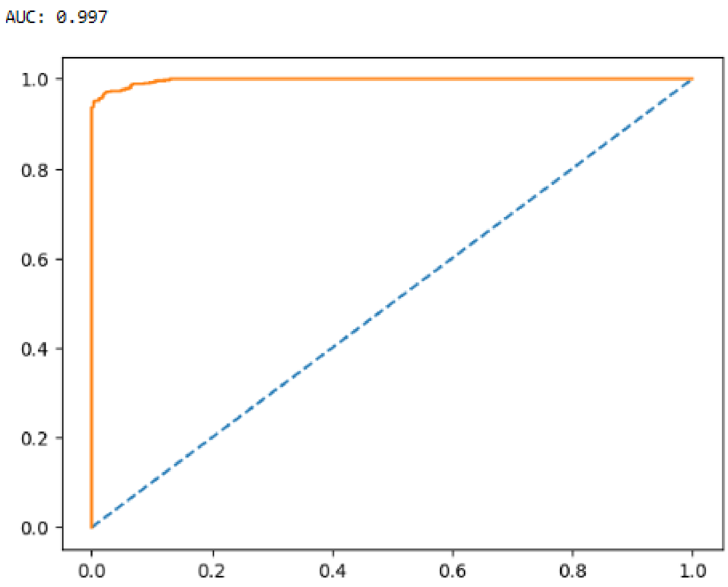


```
[[ 85  53]
 [ 36 282]]
              precision    recall  f1-score   support

           0       0.70      0.62      0.66       138
           1       0.84      0.89      0.86       318

    accuracy                           0.80       456
   macro avg       0.77      0.75      0.76       456
weighted avg       0.80      0.80      0.80       456
```

- <u>**ROC Curve & ROC_AUC score GridsearchCV:**</u>

- **Train** data:

Figure 36: Figure showing ROC curve and ROC_AUC score for the GridsearchCV Bagging Train data

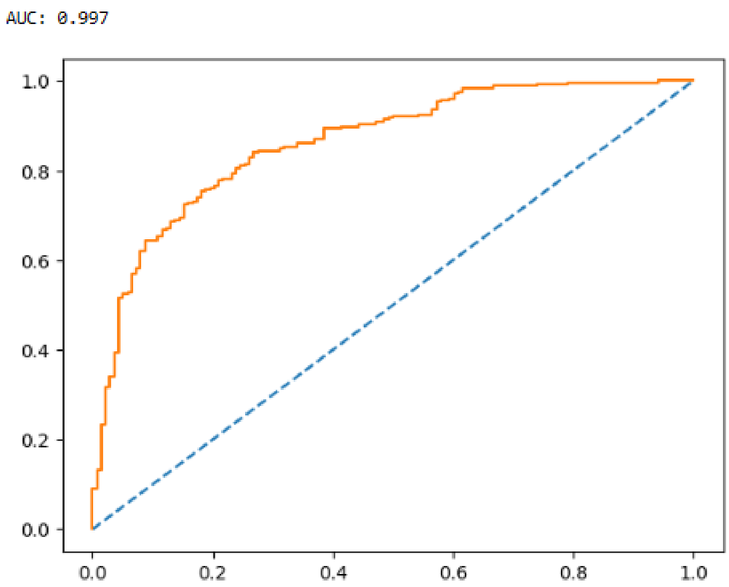


```
AUC: 0.997
```

- **ROC_AUC score for Train data is 0.997**

- **Test** data:

Figure 37: Figure showing ROC curve and ROC_AUC score for the GridsearchCV Bagging Test data



```
AUC: 0.997
```

- **ROC_AUC score for Test data is 0.997**

d) AdaBoosting:

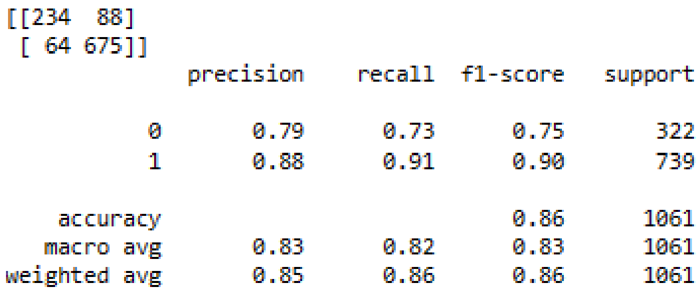
- BASE model

- Initializing and Fitting AdaBoost classifier**.**
    - **Accuracy:**
        - Accuracy score of **Train** dataset is 0.86
        - Accuracy score of **Test** dataset is 0.81

- **Confusion Matrix and Classification report:**

- **Train** data:

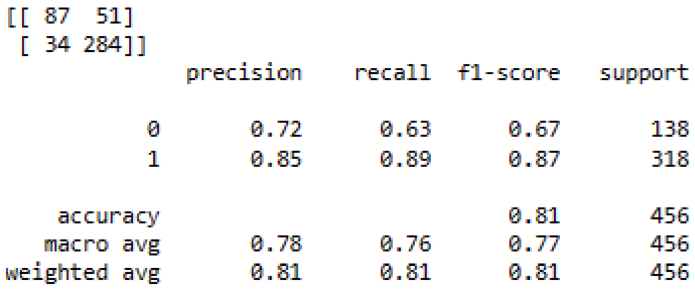
Table 85: Confusion matrix and Classification report matrix for the Train data (AdaBoost)



```
[[234  88]
 [ 64 675]]
              precision    recall  f1-score   support

           0       0.79      0.73      0.75       322
           1       0.88      0.91      0.90       739

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

- **Test** data:

Table 86: Confusion matrix and Classification report matrix for the Test data (AdaBoost)



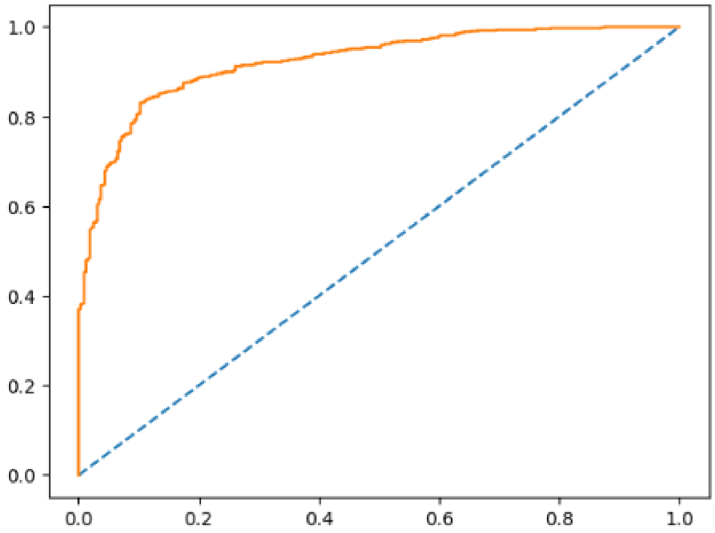
```
[[ 87  51]
 [ 34 284]]
              precision    recall  f1-score   support

           0       0.72      0.63      0.67       138
           1       0.85      0.89      0.87       318

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

- <u>**ROC Curve & ROC_AUC score:**</u>

- **Train** data:

Figure 38: Figure showing ROC curve and ROC_AUC score for the AdaBoost Train data

```
AUC: 0.923
```

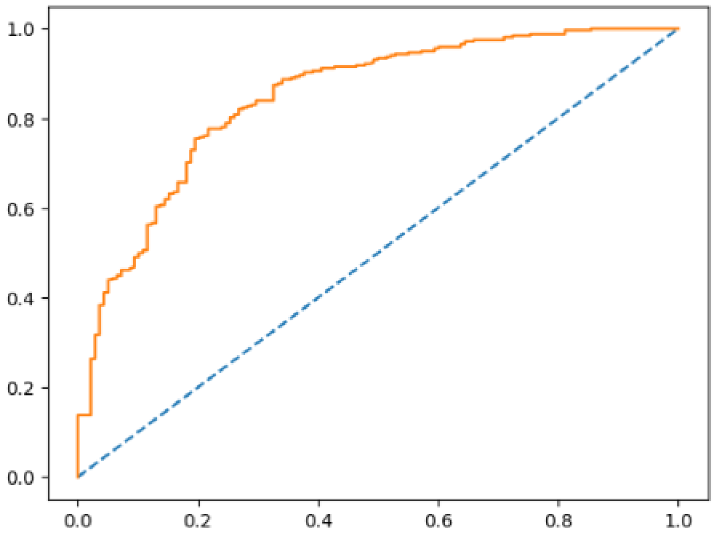


- **ROC_AUC score for Train data is 0.923**

- **Test** data:

Figure 39: Figure showing ROC curve and ROC_AUC score for the AdaBoost Test data

```
AUC: 0.923
```



- **ROC_AUC score for Test data is 0.923**

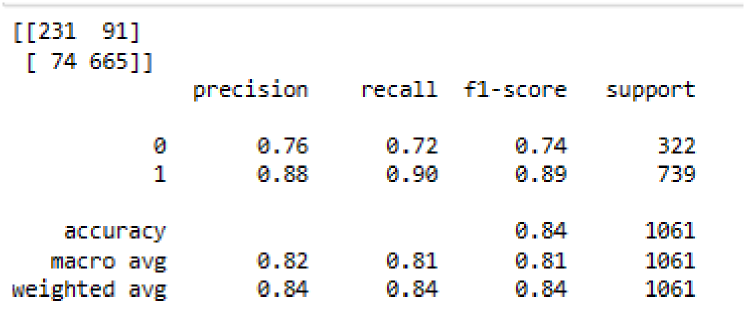
GridsearchCV for AdaBoost model

- Initializing and fitting the **AdaBoostClassifier.**
- **Accuracy:**
    - Accuracy score of **Train** dataset is 0.84
    - Accuracy score of **Test** dataset is 0.81

- **Confusion Matrix and Classification report:**

- **Train** data:

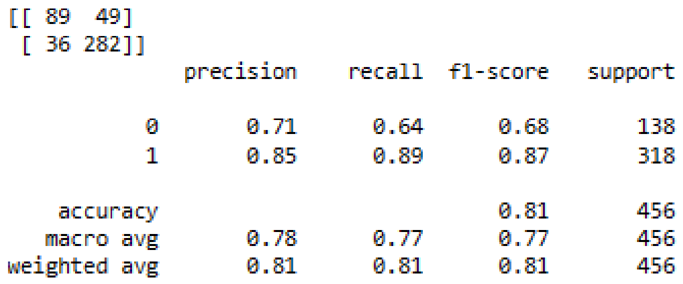
Table 87: Confusion matrix and Classification report matrix for the Train data (GridsearchCV AdaBoost model)



```
[[231  91]
 [ 74 665]]
              precision    recall  f1-score   support

           0       0.76      0.72      0.74       322
           1       0.88      0.90      0.89       739

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

- **Test** data:

Table 88: Confusion matrix and Classification report matrix for the Test data (GridsearchCV AdaBoost model)

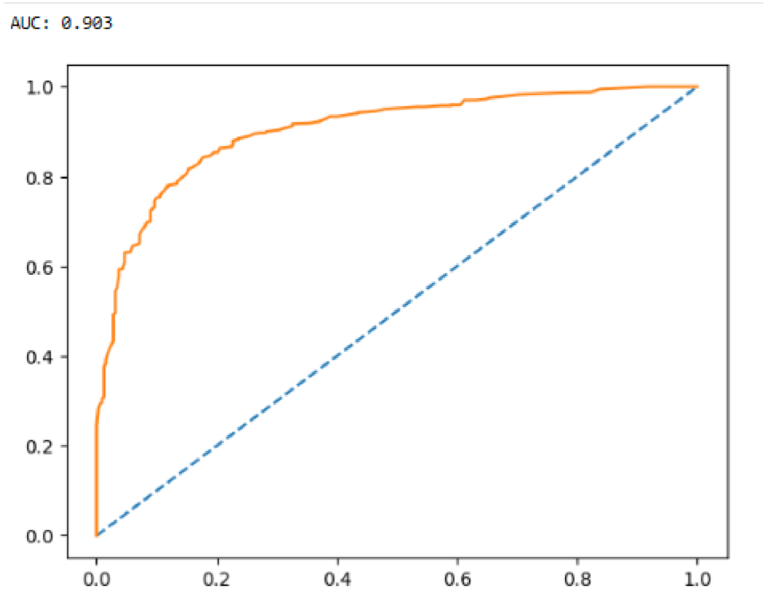


```
[[ 89  49]
 [ 36 282]]
              precision    recall  f1-score   support

           0       0.71      0.64      0.68       138
           1       0.85      0.89      0.87       318

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

- **<u>ROC Curve & ROC_AUC score GridsearchCV:</u>**

- **Train** data:

Figure 40: Figure showing ROC curve and ROC_AUC score for the GridsearchCV AdaBoost Train data

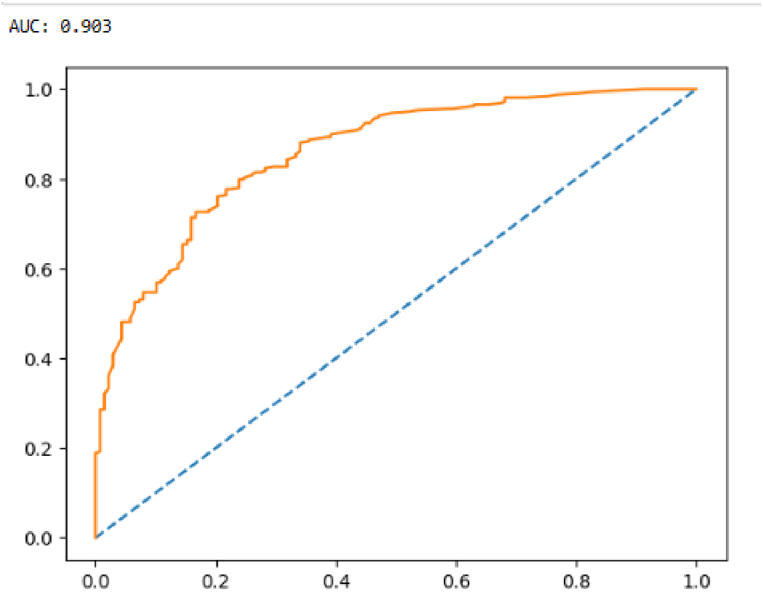


```
AUC: 0.903
```

- **ROC_AUC score for Train data is 0.903**

- **Test** data:

Figure 41: Figure showing ROC curve and ROC_AUC score for the GridsearchCV AdaBoost Test data



```
AUC: 0.903
```

- **ROC_AUC score for Test data is 0.903**

e) GradientBoost:

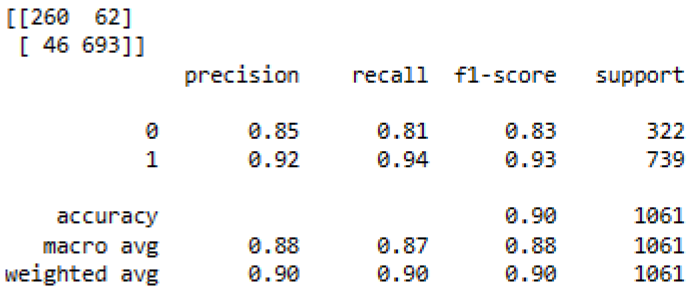
- BASE model

- Initializing and Fitting GardientBoost classifier**.**
    - **Accuracy:**
        - Accuracy score of **Train** dataset is 0.90
        - Accuracy score of **Test** dataset is 0.80

- **Confusion Matrix and Classification report:**

- **Train** data:

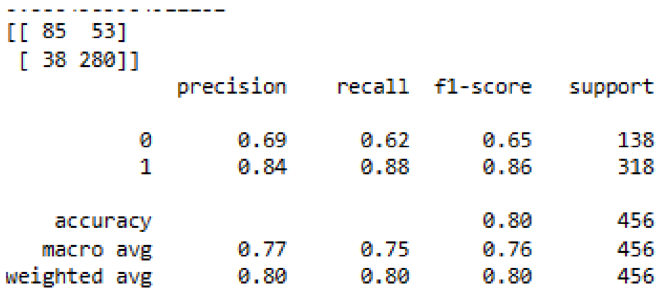
Table 89: Confusion matrix and Classification report matrix for the Train data (GradientBoost)



```
[[260  62]
 [ 46 693]]
              precision    recall  f1-score   support

           0       0.85      0.81      0.83       322
           1       0.92      0.94      0.93       739

    accuracy                           0.90      1061
   macro avg       0.88      0.87      0.88      1061
weighted avg       0.90      0.90      0.90      1061
```

- **Test** data:

Table 90: Confusion matrix and Classification report matrix for the Test data (GradientBoost)



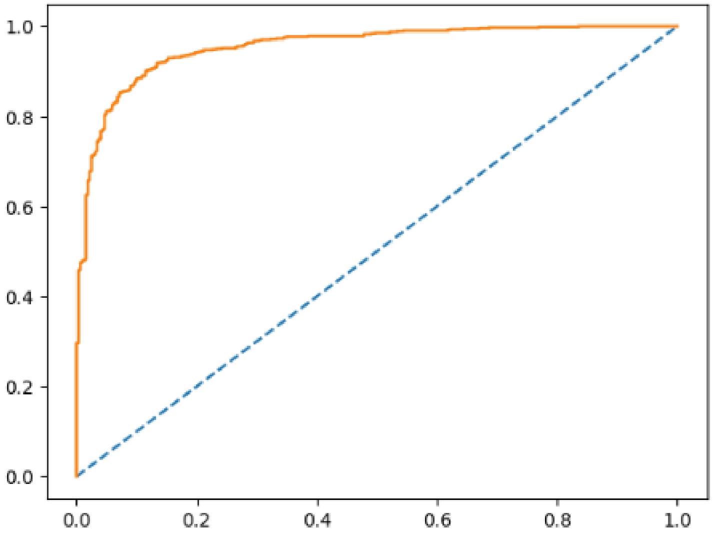
```
[[ 85  53]
 [ 38 280]]
              precision    recall  f1-score   support

           0       0.69      0.62      0.65       138
           1       0.84      0.88      0.86       318

    accuracy                           0.80       456
   macro avg       0.77      0.75      0.76       456
weighted avg       0.80      0.80      0.80       456
```

- **<u>ROC Curve & ROC_AUC score:</u>**

- **Train** data:

Figure 42: Figure showing ROC curve and ROC_AUC score for the GradientBoost Train data

```
AUC: 0.955
```

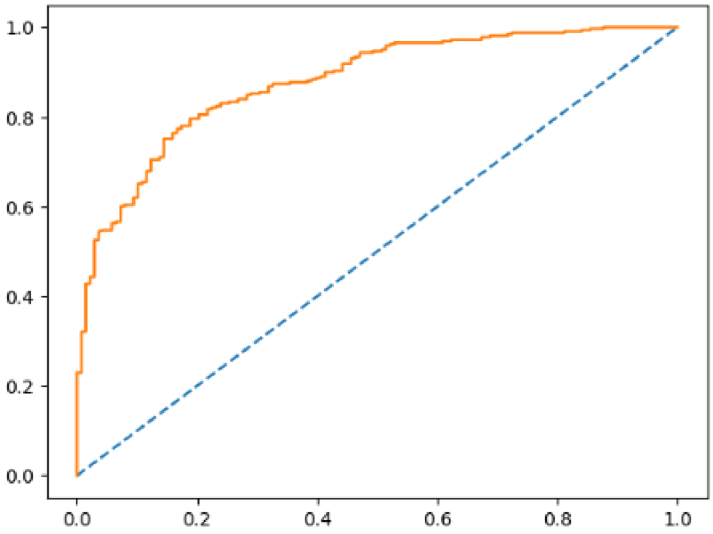


- **ROC_AUC score for Train data is 0.955**

- **Test** data:

Figure 43: Figure showing ROC curve and ROC_AUC score for the GradientBoost Test data

```
AUC: 0.955
```



- **ROC_AUC score for Test data is 0.955**

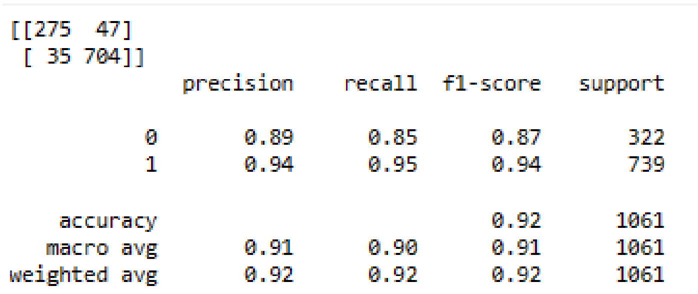
GridsearchCV for GradientBoosting model

- Initializing and fitting the **GradientBoostingClassifier.**
- **Accuracy:**
  - Accuracy score of **Train** dataset is 0.92
  - Accuracy score of **Test** dataset is 0.80

- **Confusion Matrix and Classification report:**

- **Train** data:

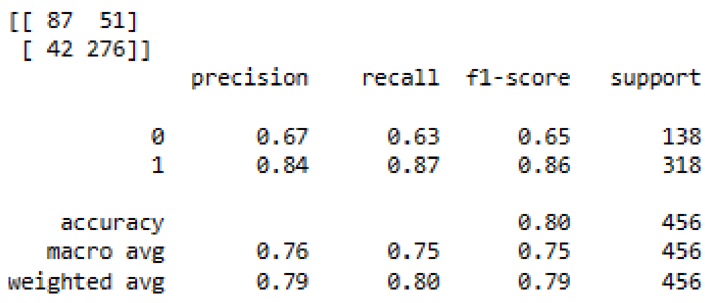
Table 91: Confusion matrix and Classification report matrix for the Train data (GridsearchCV GradientBoost model)



```
[[275  47]
 [ 35 704]]
              precision    recall  f1-score   support

           0       0.89      0.85      0.87       322
           1       0.94      0.95      0.94       739

    accuracy                           0.92      1061
   macro avg       0.91      0.90      0.91      1061
weighted avg       0.92      0.92      0.92      1061
```

- **Test** data:

Table 92: Confusion matrix and Classification report matrix for the Test data (GridsearchCV GradientBoost model)

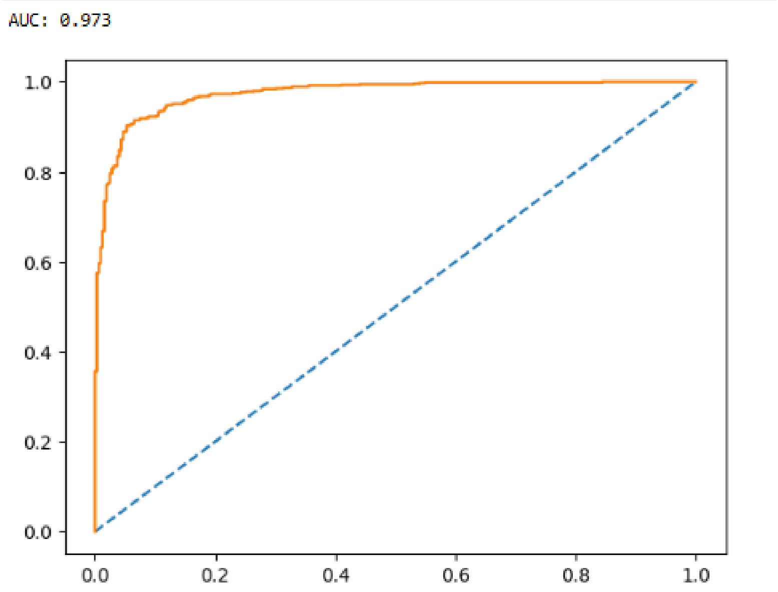


```
[[ 87  51]
 [ 42 276]]
              precision    recall  f1-score   support

           0       0.67      0.63      0.65       138
           1       0.84      0.87      0.86       318

    accuracy                           0.80       456
   macro avg       0.76      0.75      0.75       456
weighted avg       0.79      0.80      0.79       456
```

- **<u>ROC Curve & ROC_AUC score GridsearchCV:</u>**

- **Train** data:

Figure 44: Figure showing ROC curve and ROC_AUC score for the GridsearchCV GradientBoost Train data

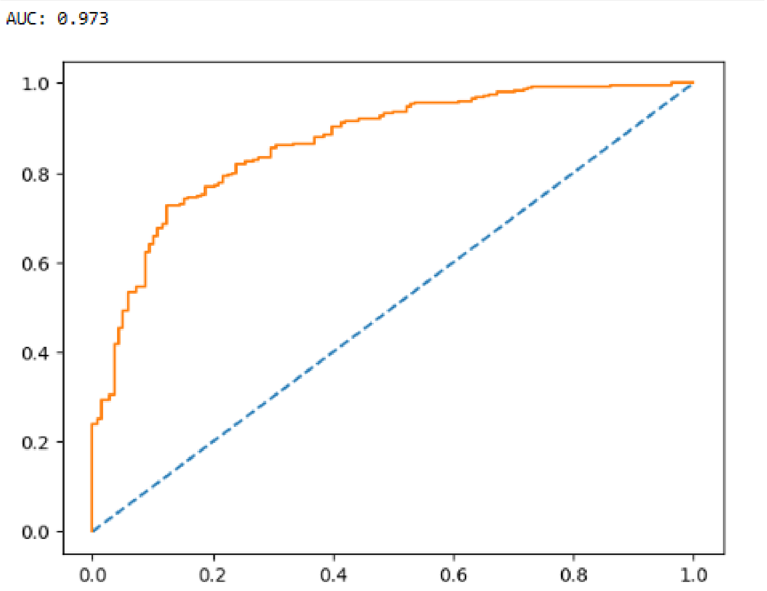


```
AUC: 0.973
```

- **ROC_AUC score for Train data is 0.973**

- **Test** data:

Figure 45: Figure showing ROC curve and ROC_AUC score for the GridsearchCV GradientBoost Test data



```
AUC: 0.973
```

- **ROC_AUC score for Test data is 0.973**

f) Logistic Regression:

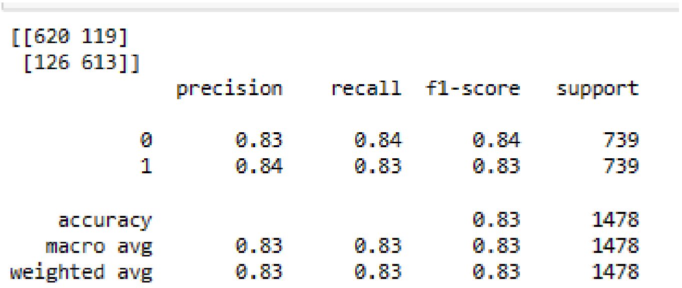
- BASE model

- Initializing and Fitting Logistic Regression classifier**.**
    - **Accuracy:**
        - Accuracy score of **Train** dataset is 0.83
        - Accuracy score of **Test** dataset is 0.80

- **Confusion Matrix and Classification report:**

- **Train** data:

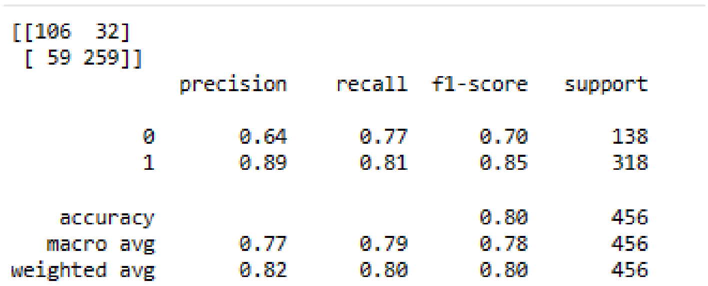
Table 93: Confusion matrix and Classification report matrix for the Train data (LogisticRegression)



```
[[620 119]
 [126 613]]
              precision    recall  f1-score   support

           0       0.83      0.84      0.84       739
           1       0.84      0.83      0.83       739

    accuracy                           0.83      1478
   macro avg       0.83      0.83      0.83      1478
weighted avg       0.83      0.83      0.83      1478
```

- **Test** data:

Table 94: Confusion matrix and Classification report matrix for the Test data (LogisticRegression)

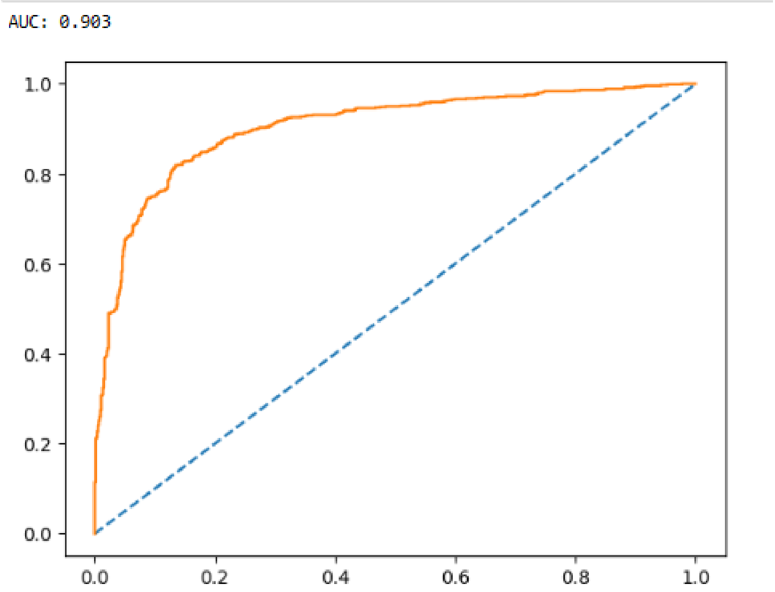


```
[[106  32]
 [ 59 259]]
              precision    recall  f1-score   support

           0       0.64      0.77      0.70       138
           1       0.89      0.81      0.85       318

    accuracy                           0.80       456
   macro avg       0.77      0.79      0.78       456
weighted avg       0.82      0.80      0.80       456
```

- <u>**ROC Curve & ROC_AUC score:**</u>

- **Train** data:

  Figure 46: Figure showing ROC curve and ROC_AUC score for the LogisticRegression Train data

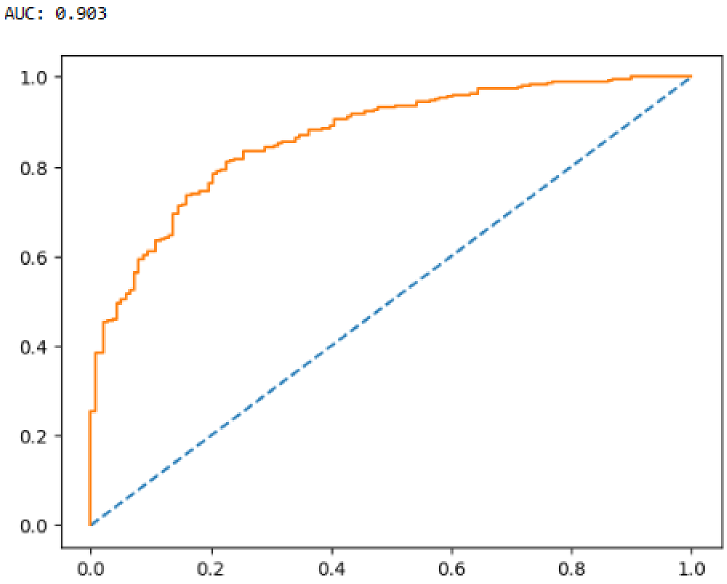


```
AUC: 0.903
```

- **ROC_AUC score for Train data is 0.903**

- **Test** data:

  Figure 47: Figure showing ROC curve and ROC_AUC score for the LogisticRegression Test data



```
AUC: 0.903
```

- **ROC_AUC score for Test data is 0.903**

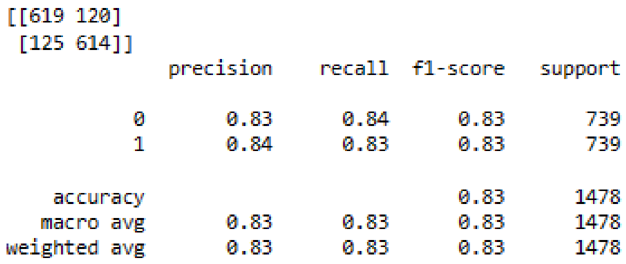
GridsearchCV for LogisticRegression model

- Initializing and fitting the **LogisticRegression Classifier.**
- **Accuracy:**
    - Accuracy score of **Train** dataset is 0.83
    - Accuracy score of **Test** dataset is 0.80

- **Confusion Matrix and Classification report:**

- **Train** data:

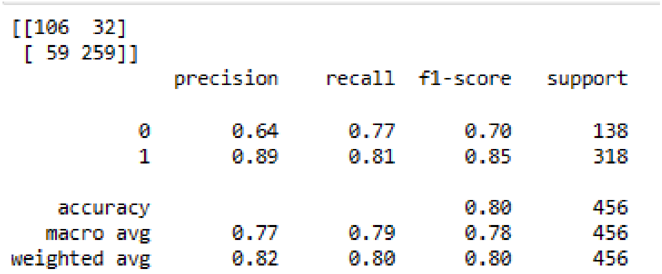
Table 95: Confusion matrix and Classification report matrix for the Train data (GridsearchCV LogisticRegression model)



```
[[619 120]
 [125 614]]
              precision    recall  f1-score   support

           0       0.83      0.84      0.83       739
           1       0.84      0.83      0.83       739

    accuracy                           0.83      1478
   macro avg       0.83      0.83      0.83      1478
weighted avg       0.83      0.83      0.83      1478
```

- **Test** data:

Table 96: Confusion matrix and Classification report matrix for the Test data (GridsearchCV LogisticRegression model)



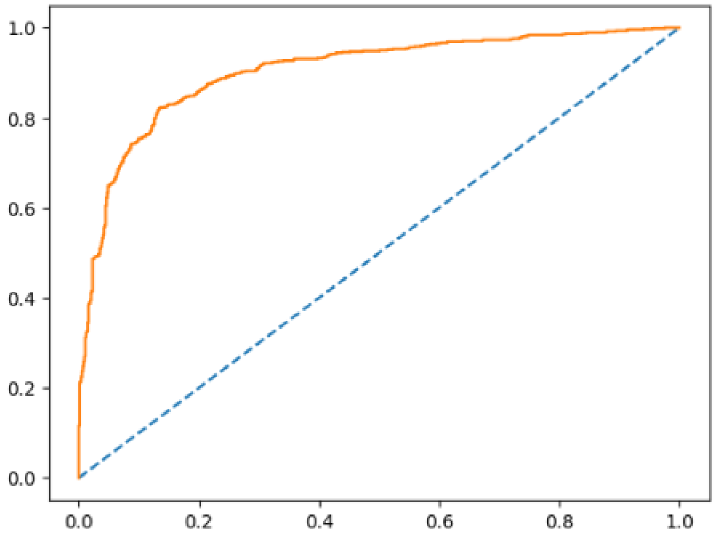
```
[[106  32]
 [ 59 259]]
              precision    recall  f1-score   support

           0       0.64      0.77      0.70       138
           1       0.89      0.81      0.85       318

    accuracy                           0.80       456
   macro avg       0.77      0.79      0.78       456
weighted avg       0.82      0.80      0.80       456
```

- **<u>ROC Curve & ROC_AUC score GridsearchCV:</u>**

- **Train** data:

Figure 48: Figure showing ROC curve and ROC_AUC score for the GridsearchCV LogisticRegression Train data

```
AUC: 0.903
```

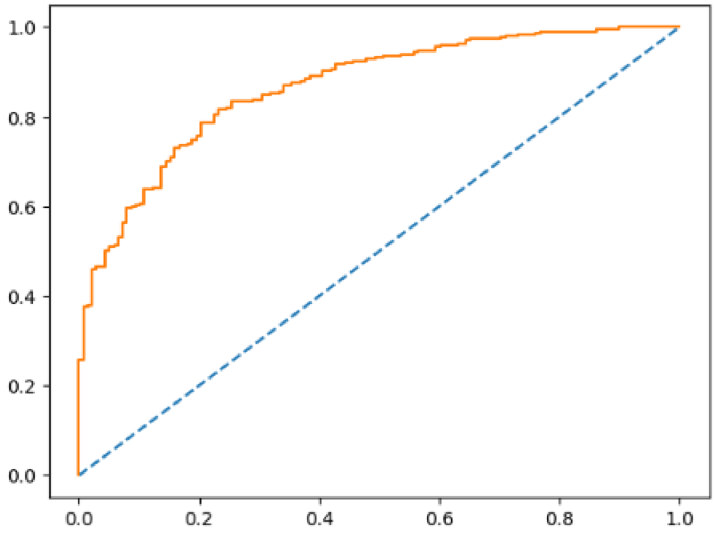


- **ROC_AUC score for Train data is 0.903**

- **Test** data:

Figure 49: Figure showing ROC curve and ROC_AUC score for the GridsearchCV LogisticRegression Test data

```
AUC: 0.903
```



- **ROC_AUC score for Test data is 0.903**

g) LDA:

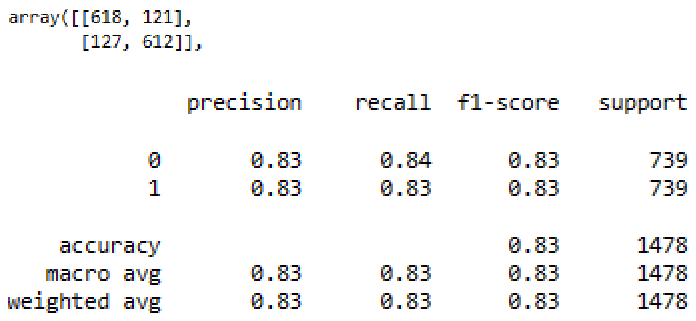
- BASE model

- Initializing and Fitting LDA classifier.
    - **Accuracy:**
        - Accuracy score of **Train** dataset is 0.83
        - Accuracy score of **Test** dataset is 0.80

- **Confusion Matrix and Classification report:**

- **Train** data:

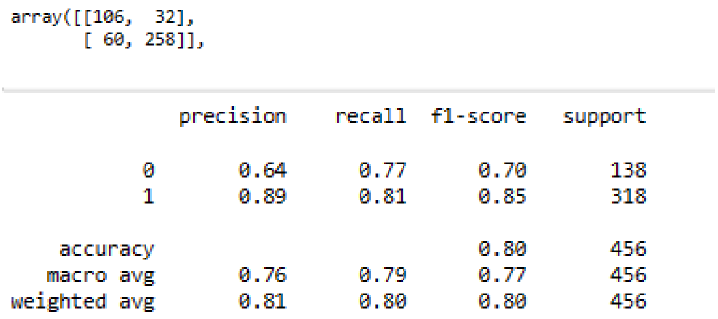
Table 97: Confusion matrix and Classification report matrix for the Train data (LDA)



```
array([[618, 121],
       [127, 612]],

              precision    recall  f1-score   support

           0       0.83      0.84      0.83       739
           1       0.83      0.83      0.83       739

    accuracy                           0.83      1478
   macro avg       0.83      0.83      0.83      1478
weighted avg       0.83      0.83      0.83      1478
```

- **Test** data:

Table 98: Confusion matrix and Classification report matrix for the Test data (LDA)

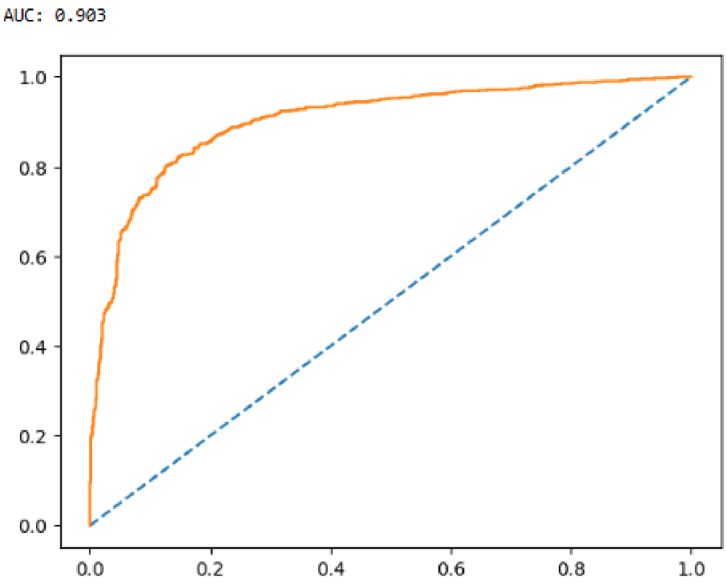


```
array([[106,  32],
       [ 60, 258]],

              precision    recall  f1-score   support

           0       0.64      0.77      0.70       138
           1       0.89      0.81      0.85       318

    accuracy                           0.80       456
   macro avg       0.76      0.79      0.77       456
weighted avg       0.81      0.80      0.80       456
```

- **<u>ROC Curve & ROC_AUC score:</u>**

- **Train** data:

Figure 50: Figure showing ROC curve and ROC_AUC score for the LDA Train data

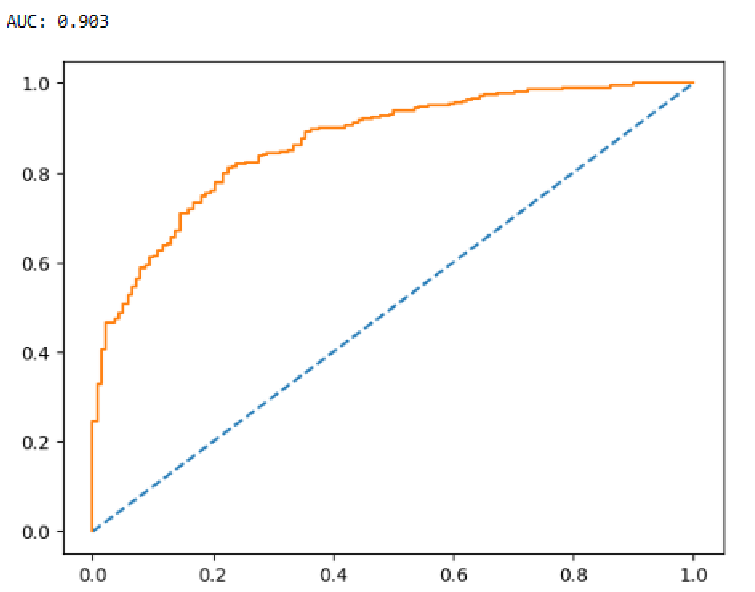


```
AUC: 0.903
```

- **ROC_AUC score for Train data is 0.903**

- **Test** data:

Figure 51: Figure showing ROC curve and ROC_AUC score for the LDA Test data



```
AUC: 0.903
```

- **ROC_AUC score for Test data is 0.903**

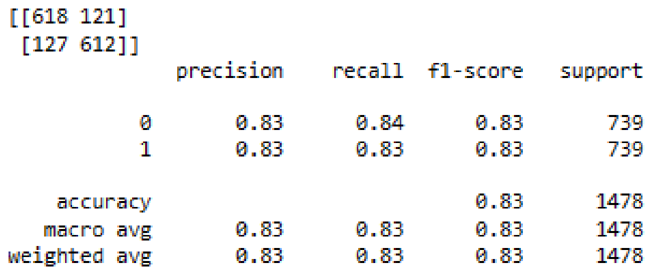
GridsearchCV for LDA model

- Initializing and fitting the **LinearDiscriminantAnalysis Classifier.**
- **Accuracy:**
    - Accuracy score of **Train** dataset is 0.83
    - Accuracy score of **Test** dataset is 0.80

- **Confusion Matrix and Classification report:**

- **Train** data:

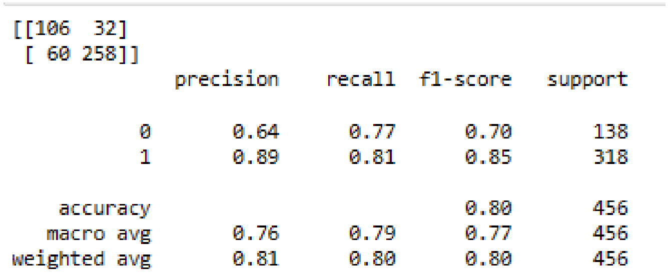
  Table 99: Confusion matrix and Classification report matrix for the Train data (GridsearchCV LDA model)



```
[[618 121]
 [127 612]]
              precision    recall  f1-score   support

           0       0.83      0.84      0.83       739
           1       0.83      0.83      0.83       739

    accuracy                           0.83      1478
   macro avg       0.83      0.83      0.83      1478
weighted avg       0.83      0.83      0.83      1478
```

- **Test** data:

  Table 100: Confusion matrix and Classification report matrix for the Test data (GridsearchCV LDA model)

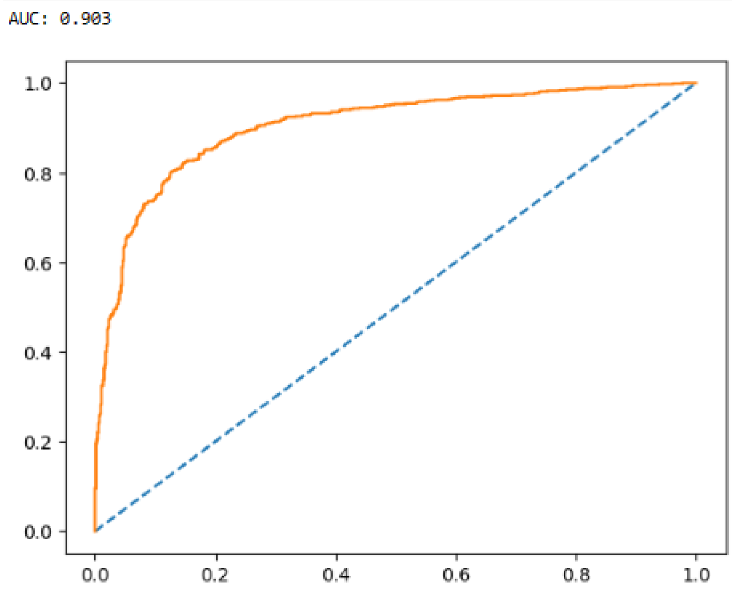


```
[[106  32]
 [ 60 258]]
              precision    recall  f1-score   support

           0       0.64      0.77      0.70       138
           1       0.89      0.81      0.85       318

    accuracy                           0.80       456
   macro avg       0.76      0.79      0.77       456
weighted avg       0.81      0.80      0.80       456
```

- **<u>ROC Curve & ROC_AUC score GridsearchCV:</u>**

- **Train** data:

Figure 52: Figure showing ROC curve and ROC_AUC score for the GridsearchCV LDA Train data

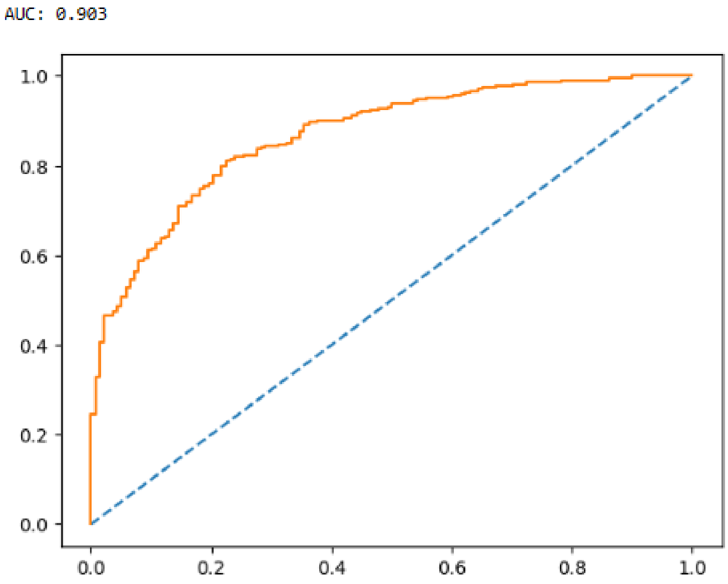


```
AUC: 0.903
```

- **ROC_AUC score for Train data is 0.903**

- **Test** data:

Figure 53: Figure showing ROC curve and ROC_AUC score for the GridsearchCV LDA Test data



```
AUC: 0.903
```

- **ROC_AUC score for Test data is 0.903**

h) Naïve Bayes:

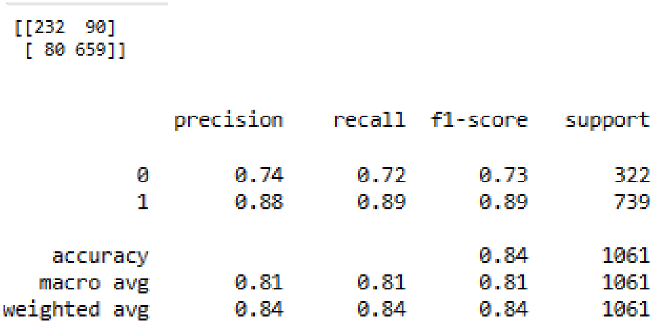
- BASE model

- Initializing and Fitting GaussianNB classifier**.**
    - **Accuracy:**
        - Accuracy score of **Train** dataset is 0.84
        - Accuracy score of **Test** dataset is 0.83

- **Confusion Matrix and Classification report:**

- **Train** data:

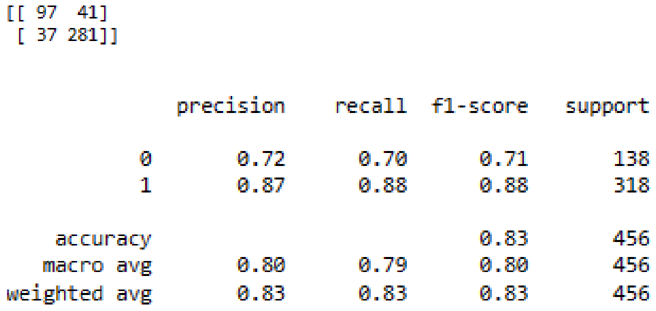
Table 101: Confusion matrix and Classification report matrix for the Train data (Naïve Bayes)



```
[[232  90]
 [ 80 659]]

              precision    recall  f1-score   support

           0       0.74      0.72      0.73       322
           1       0.88      0.89      0.89       739

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

- **Test** data:

Table 102: Confusion matrix and Classification report matrix for the Test data (Naïve Bayes)

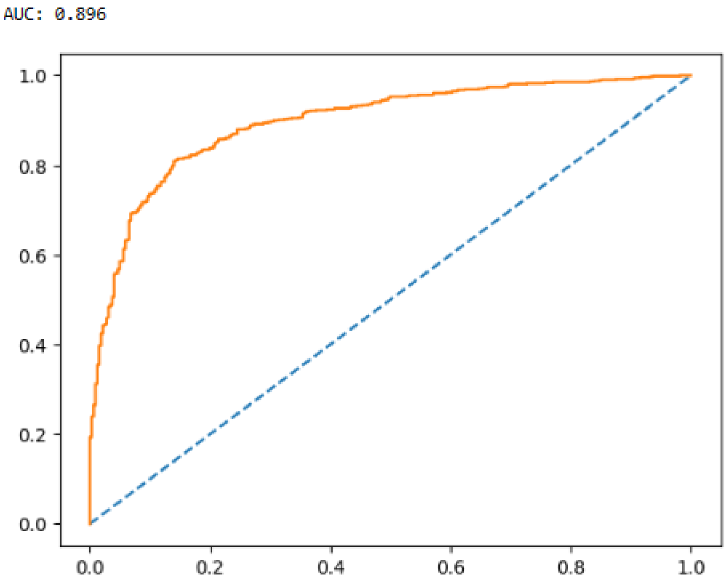


```
[[ 97  41]
 [ 37 281]]

              precision    recall  f1-score   support

           0       0.72      0.70      0.71       138
           1       0.87      0.88      0.88       318

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

- **<u>ROC Curve & ROC_AUC score:</u>**

- **Train** data:

Figure 54: Figure showing ROC curve and ROC_AUC score for the Naïve Bayes Train data

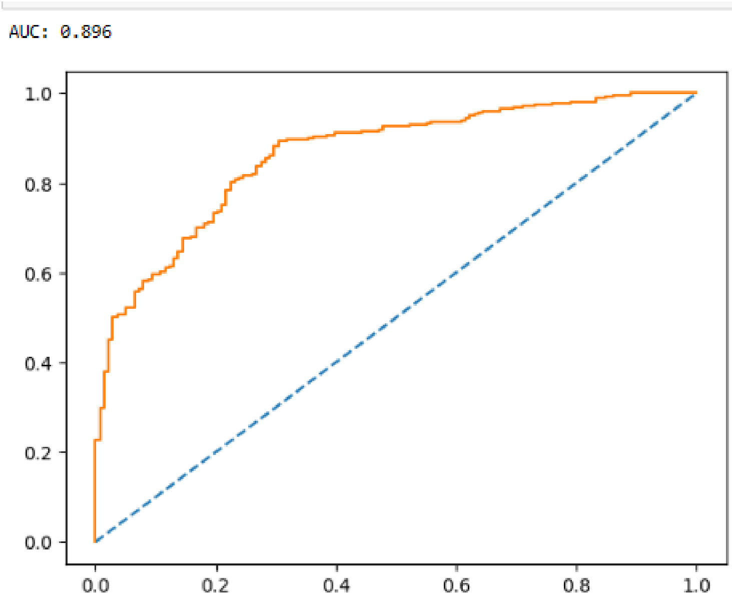


```
AUC: 0.896
```

- **ROC_AUC score for Train data is 0.896**

- **Test** data:

Figure 55: Figure showing ROC curve and ROC_AUC score for the Naïve Bayes Test data



```
AUC: 0.896
```

- **ROC_AUC score for Test data is 0.896**

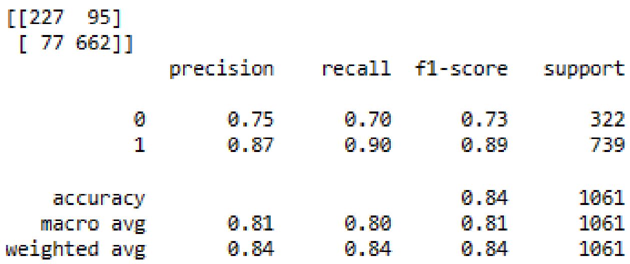
GridsearchCV for Naïve Bayes model

- Initializing and fitting the **GaussianNB Classifier.**
- **Accuracy:**
    - Accuracy score of **Train** dataset is 0.84
    - Accuracy score of **Test** dataset is 0.82

- **Confusion Matrix and Classification report:**

- **Train** data:

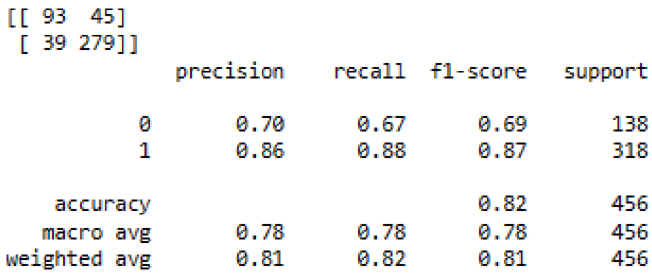
Table 103: Confusion matrix and Classification report matrix for the Train data (GridsearchCV Naïve Bayes model)



```
[[227  95]
 [ 77 662]]
              precision    recall  f1-score   support

           0       0.75      0.70      0.73       322
           1       0.87      0.90      0.89       739

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

- **Test** data:

Table 104: Confusion matrix and Classification report matrix for the Test data (GridsearchCV Naïve Bayes model)

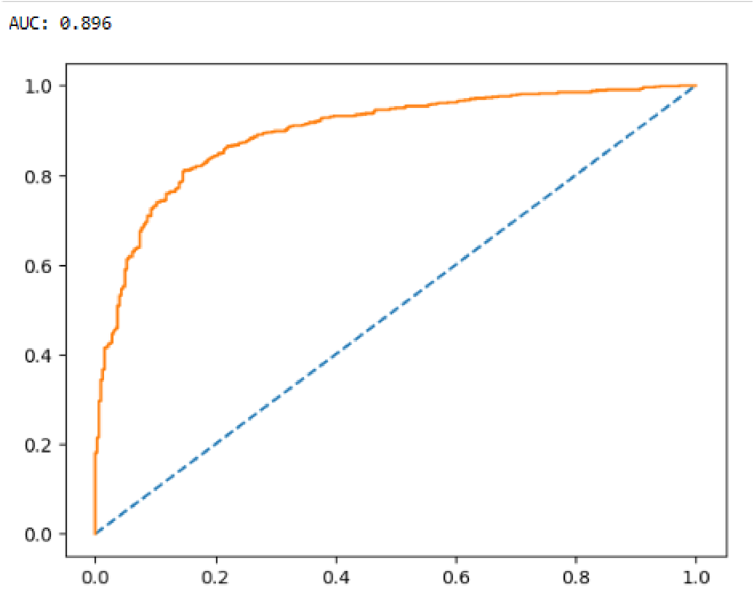


```
[[ 93  45]
 [ 39 279]]
              precision    recall  f1-score   support

           0       0.70      0.67      0.69       138
           1       0.86      0.88      0.87       318

    accuracy                           0.82       456
   macro avg       0.78      0.78      0.78       456
weighted avg       0.81      0.82      0.81       456
```

- <u>**ROC Curve & ROC_AUC score GridsearchCV:**</u>

- **Train** data:

Figure 56: Figure showing ROC curve and ROC_AUC score for the GridsearchCV Naïve Bayes Train data

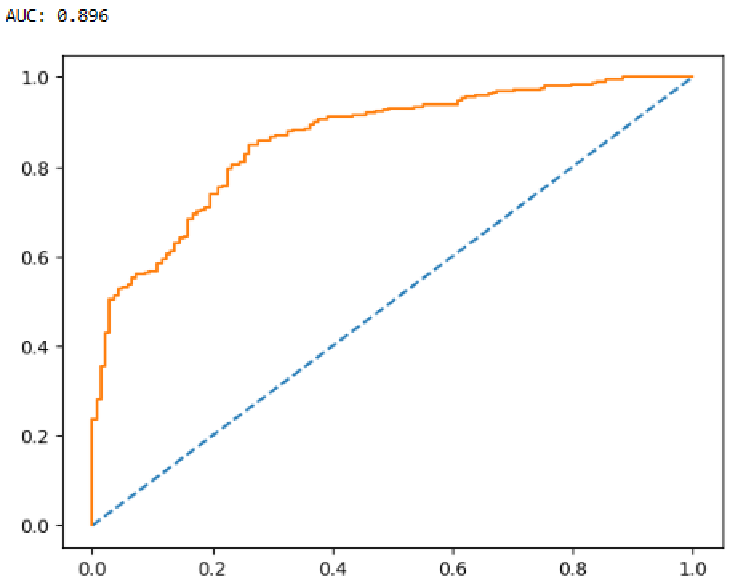


```
AUC: 0.896
```

- **ROC_AUC score for Train data is 0.896**

- **Test** data:

Figure 57: Figure showing ROC curve and ROC_AUC score for the GridsearchCV Naïve Bayes Test data



```
AUC: 0.896
```

- **ROC_AUC score for Test data is 0.896**

Table 105: Models and model performances

Models	**Model Performances**
a) KNN model	- Base model has an accuracy of 85% on Train data and 81% on test data. - Recall reduced from 0.72 to 0.64 for conservative data. - Presence of False negatives and False positives is a disadvantage. - GridsearchCV is highly overfitted as training data has 100% accuracy and test data it is reduced to 79%. - Base model is better than gridsearchcv.
b) Random Forest	- Base model is highly overfitted as training data has 100% accuracy and test data it is reduced to 79%. - Also, presence of False negatives and False positives is a disadvantage. - Base model has an accuracy of 93% on Train data and 80% on test data. - Both precision and recall of gridsearchCV is comparatively low.
c) Bagging with RandomForest	- Base model is highly overfitted as training data has 100% accuracy and test data it is reduced to 79%. - Test data reduced to 78% which was 100% on Train data. - Accuracy score of gridsearchcv on train data is 97% and test data is 80%. - GridsearchCV has low recall on conservative data.
d) AdaBoosting	- Recall of base model of conservative party on test data is 0.63 i.e., almost 37% of positive data has been falsely predicted as negatives. - Precision and recall are good for Labour party.

	- GridsearchCV has almost 74 wrongly predicted positives on train data. - Recall of GridsearchCV on conservative party data is around 64%.
e) GradientBoosting	- Base model has an accuracy of 90% on Train data and 80% on test data. - Precision and recall of conservative data are less on both train and test data. - As the base model even gridsearchCV has issues with the false negatives and false positives. - Base model is slightly better when we compare the same with the base model.
f) Logistic Regression	- SMOTE upsampling method is used to balance the data. - ROC_AUC score of base model for train data and test data is 0.903 respectively. - ROC_AUC score of gridsearchCV for train data and test data is 0.903 respectively, which is same as the base model. - Precision of base model and gridsearchCV on conservative party train data is around 64%.
g) LDA	- SMOTE upsampling method is used to balance the data. - Both the models have same accuracy score on test data. - Accuracy score for base model of train data is 83% and test data is 80%. - Accuracy score for gridsearchCV of train data is 83% and test data is 80% which is as same as base model.
h) Naïve Bayes	- Accuracy score for base model of train data is 84% and test data is 83%. - Precision and recall of base model is comparatively good.

	- GridsearchCV has low recall on the conservative party data. - Base model has a good fit between the train and test data.

Finalizing Model:

- We can go ahead with **Naïve Bayes** base model based on the accuracy score on train and test data.
- Accuracy score of **NB** base model on train data is 84% and test data is 83%.
- Model looks like a good fit and further can be improved by means such as balancing data etc.
- Also, the model has ROC_AUC score of 90%.
- Comparing with the other models, **NB** model has a good precision and recall on both train and test data (i.e., above 70% on recall and above 87% on precision), whereas other models have an issue with precision or recall on test data.

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.

Ans:

- **We are using Naïve Bayes model as it has a good accuracy score and ROC_AUC scire and goes well with prediction of data between the train and test data.**

- **Insights:**

- Majority of the voters are women when compared with men.
- Tony Blair received a four rating from the majority of voters, whereas Hague received mix of two and four rating of which majority of them gave two rating.
- About 233 voters gave 1 rating to Hague of conservative party and only 97 voters gave 1 rating to Blair of labour party. This shows the voters dissatisfaction with Hague.

- Majority of the voters who have a Eurosceptic attitude might have shifted towards Labour party.
- Majority of women and men voted for Labour party.

- **Recommendations**:

- Business being a leading news channel can also focus on educating voters who do not have any political knowledge (455 of 1525 voters have zero knowledge on politics), by discussing more about the facts, current affairs, commitments made, etc. This helps the voters to stay updated with minimal knowledge on politics and vote wisely as there is a chance for these people to switch parties when needed, which effects the accuracy of the model by having wrong predictions of the data.
- It would be great if business also collects data regarding the previous ruling party (such as satisfied or dissatisfied, commitments fulfilled or not, reasons for satisfaction or dissatisfaction etc.). This helps in analysing voters' behaviour to stay loyal or to switch the party.
- Business can also advertise elections by various campaigns i.e., through social medias and can conduct online voting in order to gain maximum crowd.

Problem 2

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

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

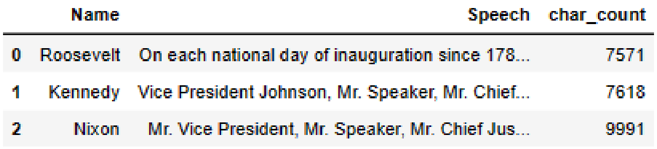
(Hint: use .words(), .raw(), .sent() for extracting counts)

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

Ans:

- Number of characters for the mentioned documents are as follows:

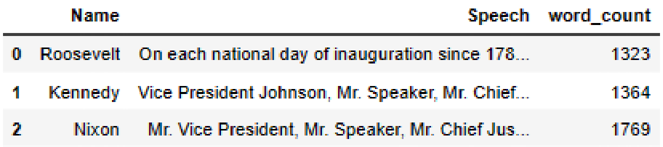
Table 106: Number of characters



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

- Number of words for the mentioned documents are as follows:

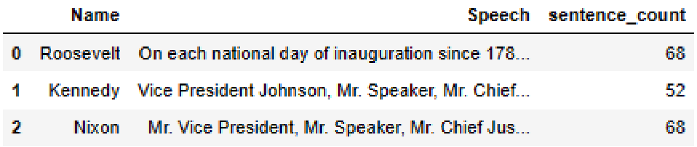
Table 107: Number of words



	Name	Speech	word_count
0	Roosevelt	On each national day of inauguration since 178...	1323
1	Kennedy	Vice President Johnson, Mr. Speaker, Mr. Chief...	1364
2	Nixon	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	1769

- Number of sentences for the mentioned documents are as follows:

Table 108: Number of sentences



	Name	Speech	sentence_count
0	Roosevelt	On each national day of inauguration since 178...	68
1	Kennedy	Vice President Johnson, Mr. Speaker, Mr. Chief...	52
2	Nixon	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	68

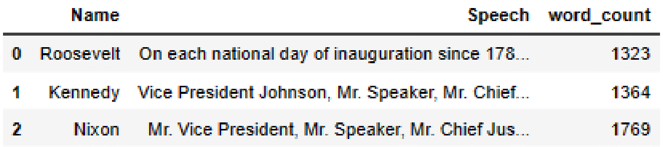
2.2 Remove all the stopwords from the three speeches. Show the word count before and after the removal of stopwords. Show a sample sentence after the removal of stopwords.

Ans:

- Removing stopwords from all the three speeches by using nltk function.
- Number of words before and after removing **Stopwords**

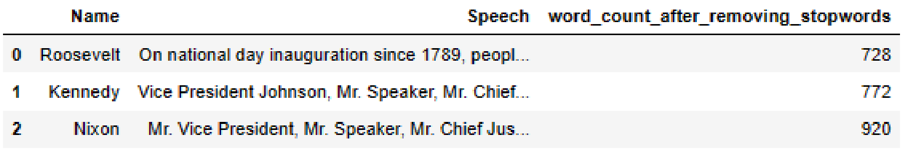
Table 109: Number of words before and after removing Stopwords

<u>Before removing Stopwords</u>



	Name	Speech	word_count
0	Roosevelt	On each national day of inauguration since 178...	1323
1	Kennedy	Vice President Johnson, Mr. Speaker, Mr. Chief...	1364
2	Nixon	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	1769

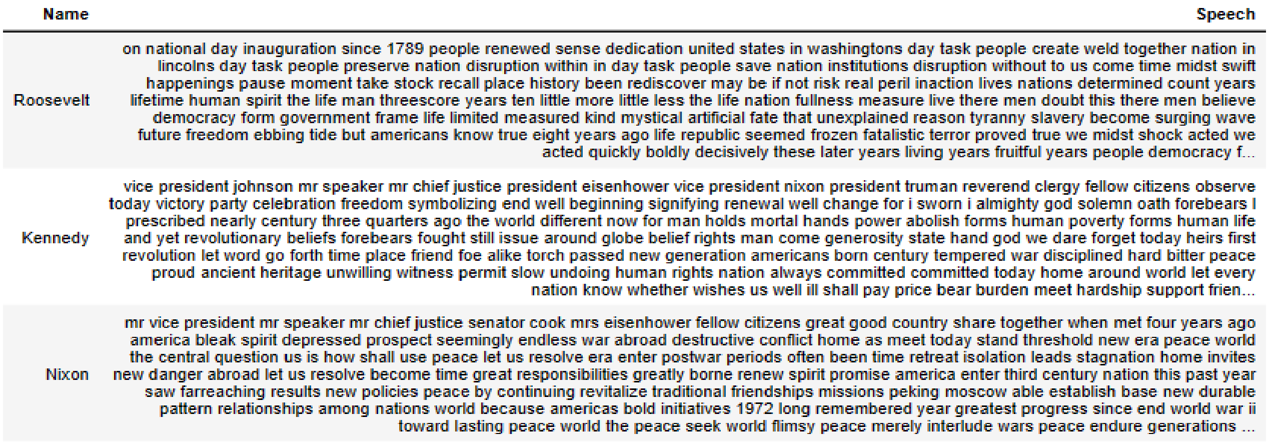
After removing Stopwords



	Name	Speech	word_count_after_removing_stopwords
0	Roosevelt	On national day inauguration since 1789, peopl...	728
1	Kennedy	Vice President Johnson, Mr. Speaker, Mr. Chief...	772
2	Nixon	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	920

- Few stopwords we noted in the speeches are ‘each’, ‘of’, ‘the’, ‘have’, ‘to’, ‘and’, ‘was’, ‘from’, ‘a’, ‘in’, etc…
- Sample sentences after removing Stopwords

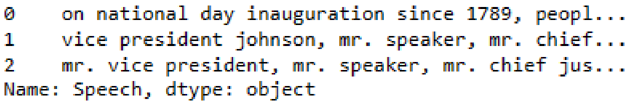
Table 110: Sample sentences



Name	Speech
Roosevelt	on national day inauguration since 1789 people renewed sense dedication united states in washingtons day task people create weld together nation in lincolns day task people preserve nation disruption within in day task people save nation institutions disruption without to us come time midst swift happenings pause moment take stock recall place history been rediscover may be if not risk real peril inaction lives nations determined count years lifetime human spirit the life man threescore years ten little more little less the life nation fullness measure live there men doubt this there men believe democracy form government frame life limited measured kind mystical artificial fate that unexplained reason tyranny slavery become surging wave future freedom ebbing tide but americans know true eight years ago life republic seemed frozen fatalistic terror proved true we midst shock acted we acted quickly boldly decisively these later years living years fruitful years people democracy f...
Kennedy	vice president johnson mr speaker mr chief justice president eisenhower vice president nixon president truman reverend clergy fellow citizens observe today victory party celebration freedom symbolizing end well beginning signifying renewal well change for i sworn i almighty god solemn oath forebears I prescribed nearly century three quarters ago the world different now for man holds mortal hands power abolish forms human poverty forms human life and yet revolutionary beliefs forebears fought still issue around globe belief rights man come generosity state hand god we dare forget today heirs first revolution let word go forth time place friend foe alike torch passed new generation americans born century tempered war disciplined hard bitter peace proud ancient heritage unwilling witness permit slow undoing human rights nation always committed committed today home around world let every nation know whether wishes us well ill shall pay price bear burden meet hardship support frien...
Nixon	mr vice president mr speaker mr chief justice senator cook mrs eisenhower fellow citizens great good country share together when met four years ago america bleak spirit depressed prospect seemingly endless war abroad destructive conflict home as meet today stand threshold new era peace world the central question us is how shall use peace let us resolve era enter postwar periods often been time retreat isolation leads stagnation home invites new danger abroad let us resolve become time great responsibilities greatly borne renew spirit promise america enter third century nation this past year saw farreaching results new policies peace by continuing revitalize traditional friendships missions peking moscow able establish base new durable pattern relationships among nations world because americas bold initiatives 1972 long remembered year greatest progress since end world war ii toward lasting peace world the peace seek world flimsy peace merely interlude wars peace endure generations ...

- Additionally, we are also performing basic preprocessing
- **Lowercase conversion**

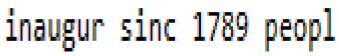
Table 111: After converting speeches to lowercase



```
0    on national day inauguration since 1789, peopl...
1    vice president johnson, mr. speaker, mr. chief...
2    mr. vice president, mr. speaker, mr. chief jus...
Name: Speech, dtype: object
```

- Also, we are **removing punctuation** and **special characters** from the mentioned speeches.
- **Stemming:** Is a process that remove suffices of words**.**

for ex:



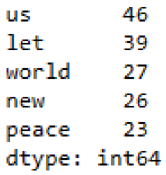
inaugur sinc 1789 peopl

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).

Ans:

- Top five words that occur the most in all three speeches are as follows:

Table 112: Top five words that occur the most

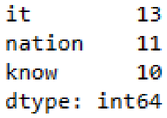


```
us       46
let      39
world    27
new      26
peace    23
dtype: int64
```

- Top three words that occur the most of all three president speeches are as follows:

- **<u>Roosevelt</u>**

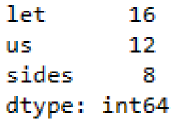
Table 113: Top three words that occur the most in Roosevelt's speech



```
it        13
nation    11
know      10
dtype: int64
```

- **<u>Kennedy</u>**

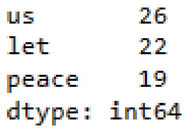
Table 114: Top three words that occur the most in Kennedy's speech



```
let      16
us       12
sides     8
dtype: int64
```

- **<u>Nixon</u>**

Table 115: Top three words that occur the most in Nixon's speech



```
us       26
let      22
peace    19
dtype: int64
```

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

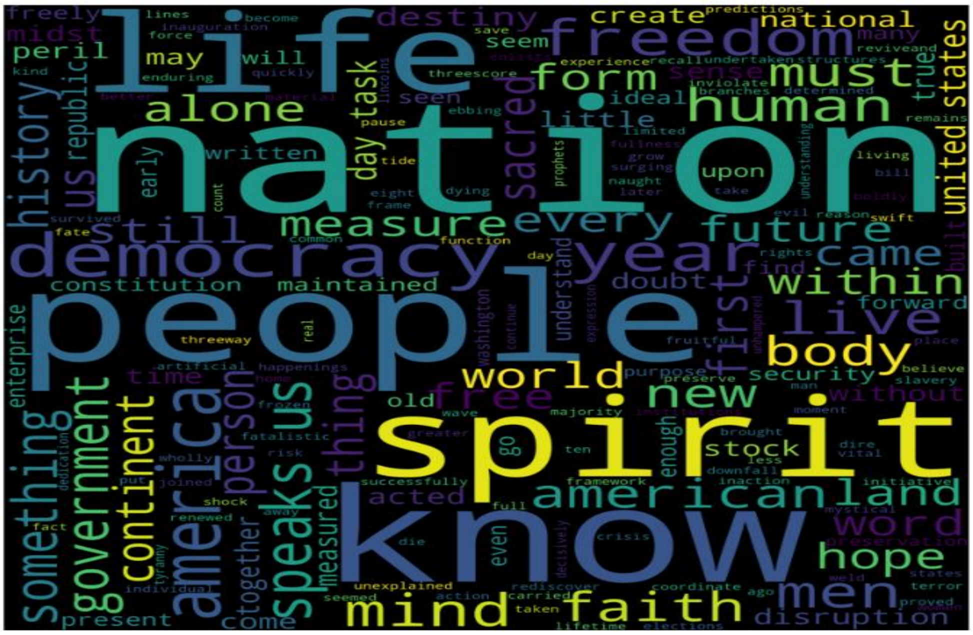
Ans:

- Word cloud of each of the three speeches are as follows:

- **Roosevelt**

Figure 58: Plot showing word cloud of Roosevelt's speech

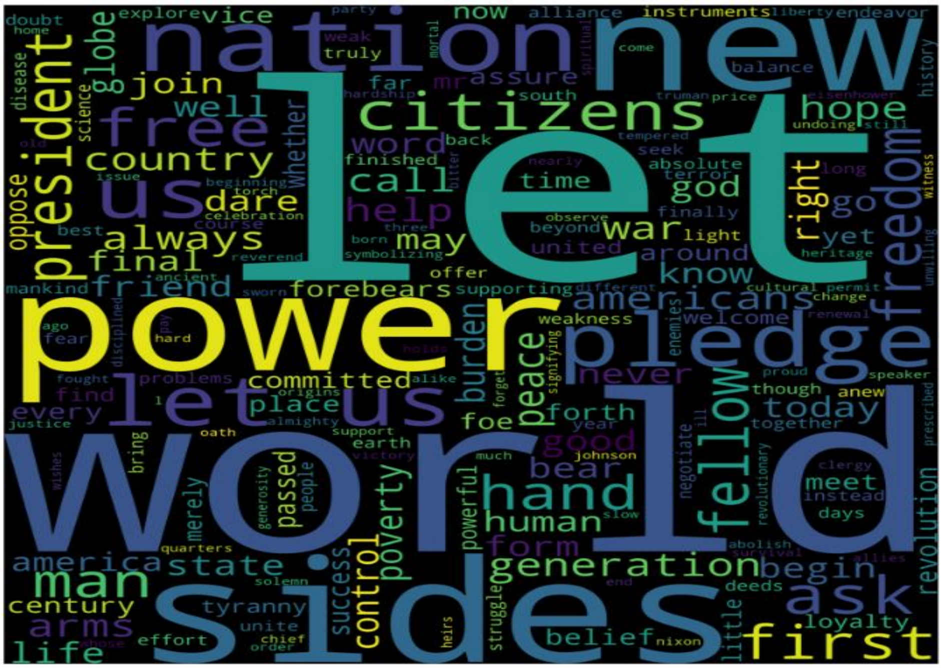
Word Cloud of Roosevelt's Speech (after cleaning)!!



- **Kennedy**

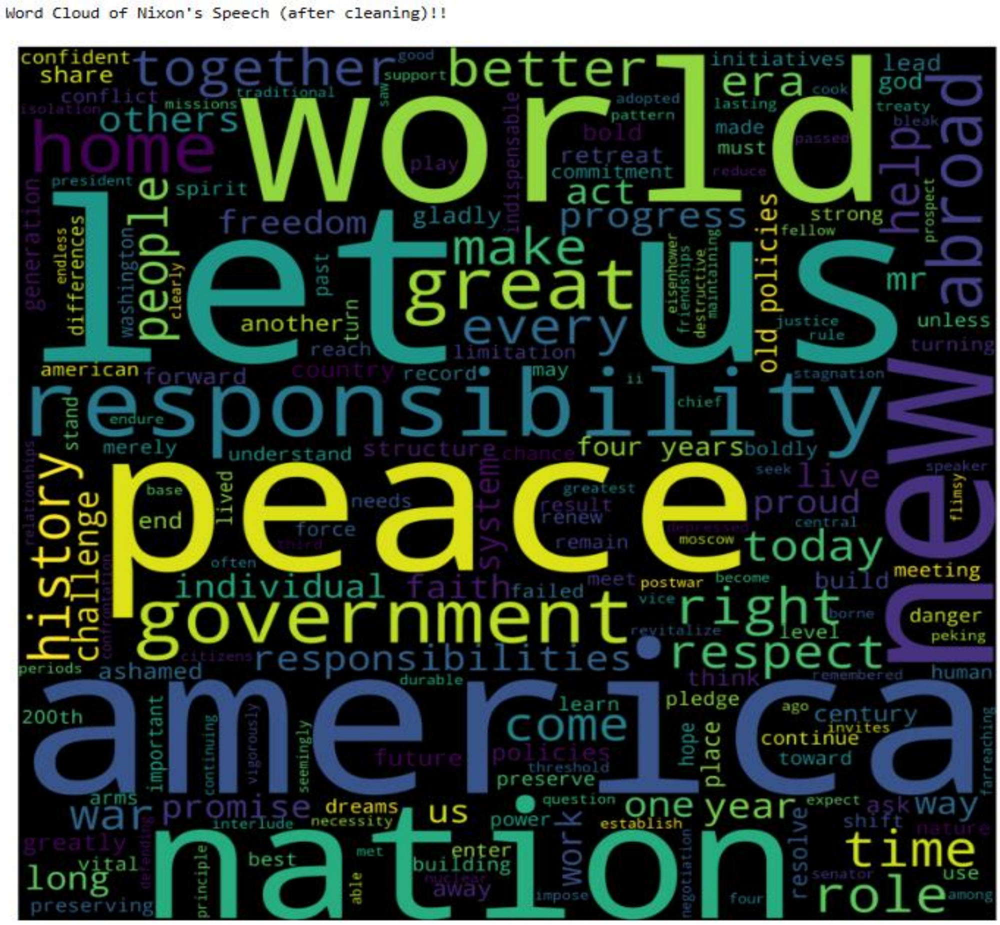
Figure 59: Plot showing word cloud of Kennedy's speech

Word Cloud of Kennedy's Speech (after cleaning)!!



- **Nixon**

Figure 60: Plot showing word cloud of Nixon’s speech



--THE END---