Problem 1:

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

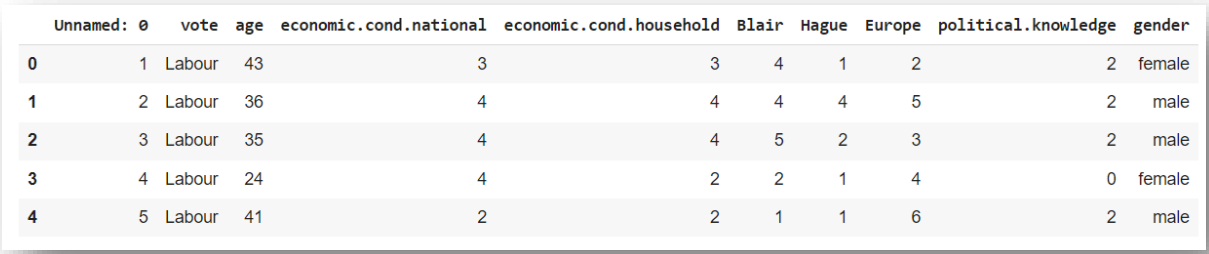
Data Dictionary:

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

Data Ingestion: 11 marks

1.1 Read the dataset. Do the descriptive statistics and do the null value condition check. Write an inference on it. (4 Marks)

Table 1: Head of the dataset showing the first 5 records

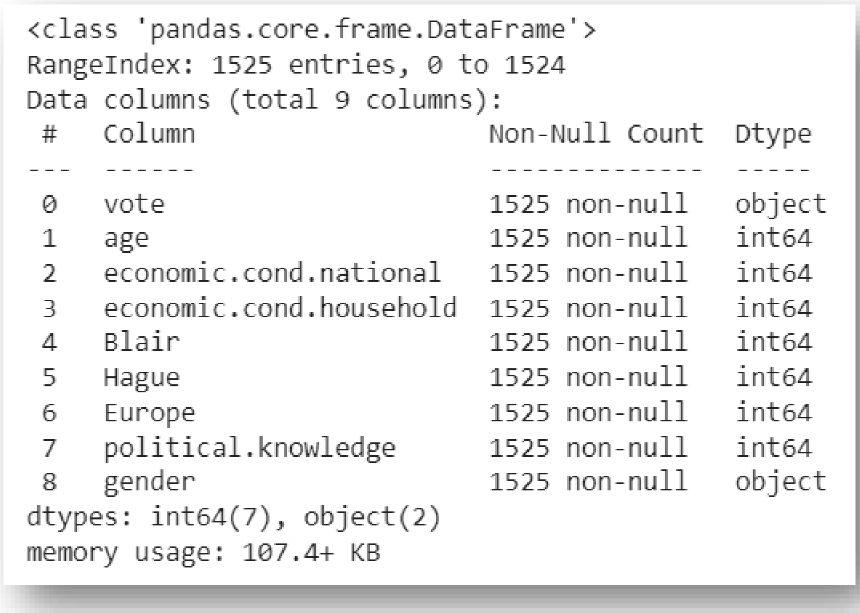


	Unnamed: 0	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	1	Labour	43	3	3	4	1	2	2	female
1	2	Labour	36	4	4	4	4	5	2	male
2	3	Labour	35	4	4	5	2	3	2	male
3	4	Labour	24	4	2	2	1	4	0	female
4	5	Labour	41	2	2	1	1	6	2	male

The dataset was loaded and the head of the dataset was checked. Table 1 shows the first 5 records of the dataset. From this table, we can see the different variables or columns of the dataset.

The 'Unnamed: 0' is basically the serial number or the index and hence it is dropped as it is of no use in the model. It may also interfere with the exploratory data analysis.

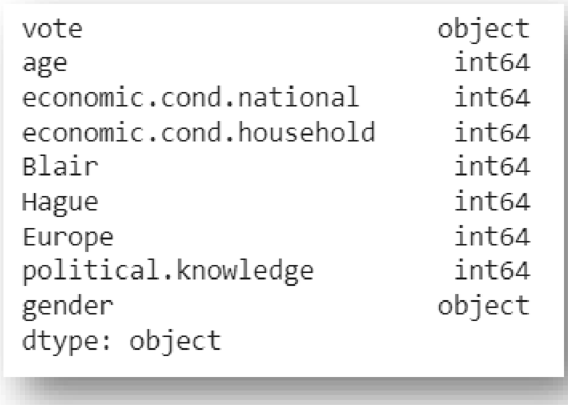
Table 2: Information of the dataset



```
<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
```

After dropping ‘Unnamed: 0’ column, the dataset has 9 columns and 1525 records seen in Table 2. There are 1525 non-null records in all the columns meaning there are no missing records based on this initial analysis that was done.

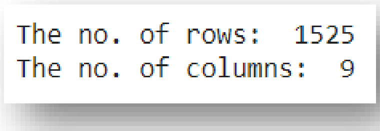
Table 3: Data type of the columns in the dataset



```
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
```

The column ‘vote’ and ‘gender’ is of object data type while all the other columns are of integer data type. This is seen in Table 1, Table 2 and Table 3. There are 8 independent variables and one target variable – ‘vote’.

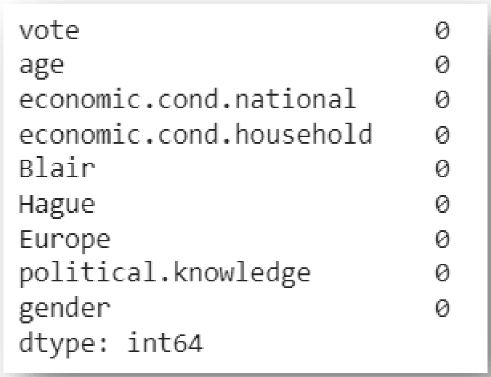
Table 4: Shape of the dataset



```
The no. of rows:  1525
The no. of columns:  9
```

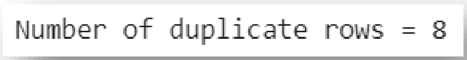
The shape of the data is (1525, 10) meaning the dataset has 1525 rows and 9 columns shown in Table 4.

Table 5: Missing value of the columns 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
```

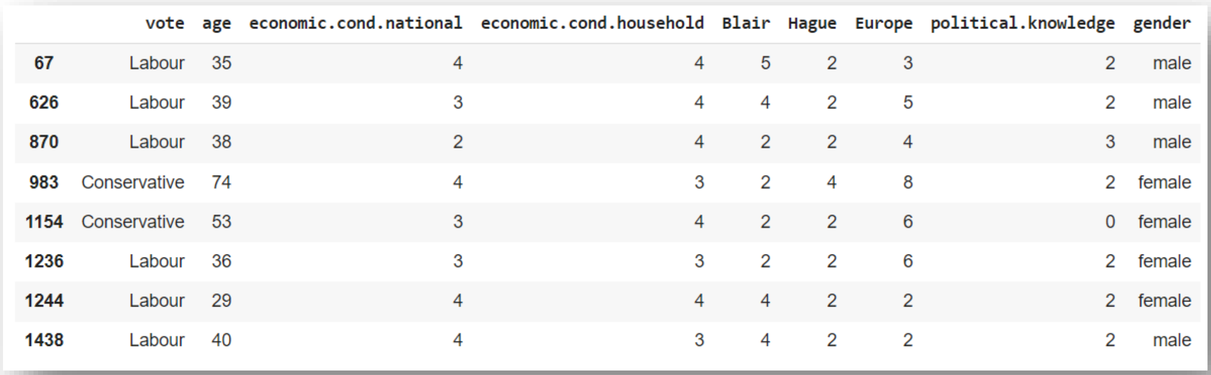
The dataset was further checked for missing values and it is seen from Table 5 that there are no missing values in the dataset.



```
Number of duplicate rows = 8
```

Figure 1: Number of duplicates in the dataset

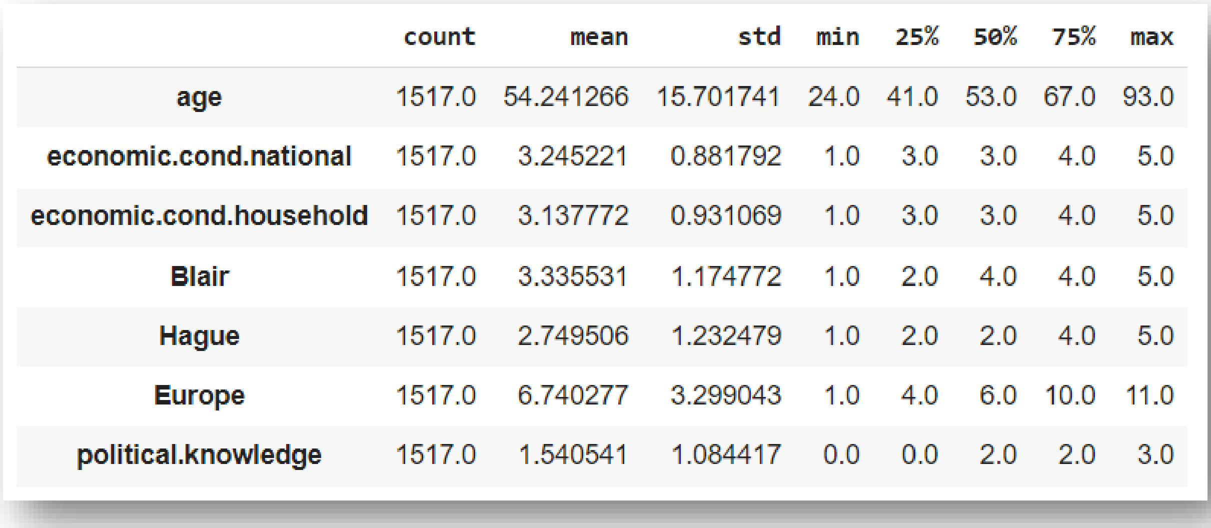
Table 6: Duplicates in the dataset



	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
67	Labour	35	4	4	5	2	3	2	male
626	Labour	39	3	4	4	2	5	2	male
870	Labour	38	2	4	2	2	4	3	male
983	Conservative	74	4	3	2	4	8	2	female
1154	Conservative	53	3	4	2	2	6	0	female
1236	Labour	36	3	3	2	2	6	2	female
1244	Labour	29	4	4	4	2	2	2	female
1438	Labour	40	4	3	4	2	2	2	male

The dataset is checked for duplicates and it was found that there 8 duplicate rows seen in Table 6 and Figure 1. The duplicates are dropped and upon dropping the duplicates, the shape of the dataset is (1517, 9) meaning the dataset now has 1517 rows and 9 columns.

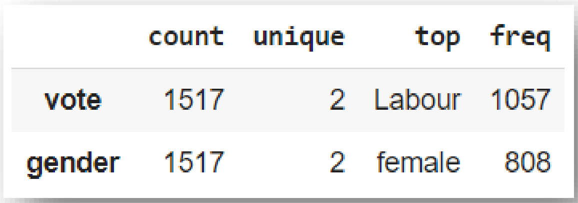
Table 7: Descriptive statistics of the numerical columns in the dataset



	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

Table 7 shows the description or the summary of the numerical columns in the dataset after dropping the ‘Unnamed: 0’ column and the duplicates in the dataset. It can be seen that all the columns in this dataframe have 1517 values. There are no missing values. By looking at Table 7, we are able to deduce that ‘age’ has the highest mean value while ‘political.knowledge’ has the lowest mean value. ‘age’ has the highest standard deviation value while ‘economic.cond.national’ has the lowest standard deviation value. This is probably because age and economic.cond.national are two different measurements. ‘age’ is in years while ‘economic.cond.national’ is the assessment of current national economic conditions ranging from 1 to 5. The mean and median values are approximately equal in all variables.

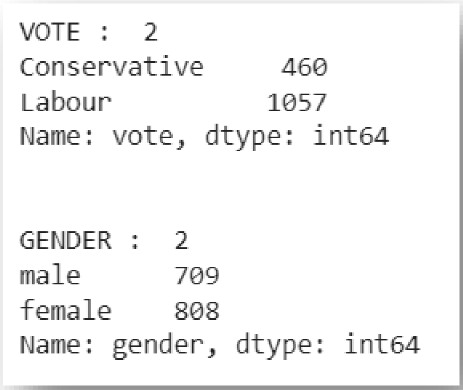
Table 8: Descriptive statistics of the categorical columns in the dataset



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

Table 8 shows the description or the summary of the categorical columns in the dataset. It can be seen that all the columns in this dataframe have 1517 values. There are no missing values.

Table 9: Unique categories and value counts of categorical columns in the dataset



```
VOTE :  2
Conservative     460
Labour          1057
Name: vote, dtype: int64


GENDER :  2
male      709
female    808
Name: gender, dtype: int64
```

Table 9 shows that there are 2 categories in each of the categorical variables and the counts of the categories.

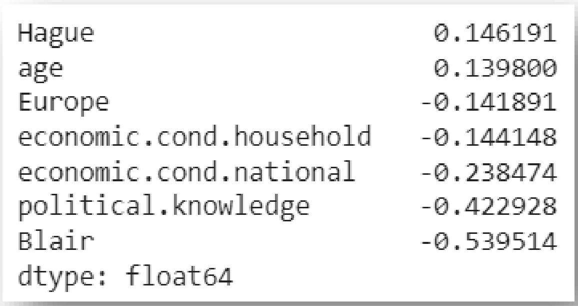
1.2 Perform Univariate and Bivariate Analysis. Do exploratory data analysis. Check for Outliers. (7 Marks)

Univariate Analysis:

Numerical variables:

The dataset was divided into numerical and categorical based on the data type of the variables. The numerical dataset was used to calculate the skewness, plot the univariate distribution and the boxplot.

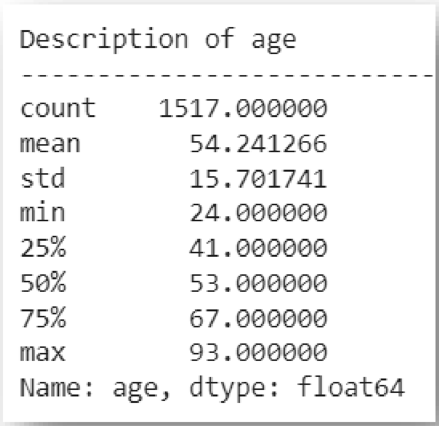
Table 10: Skewness of the variables of the dataset



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

1. Age

Table 11: Description of 'age'



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

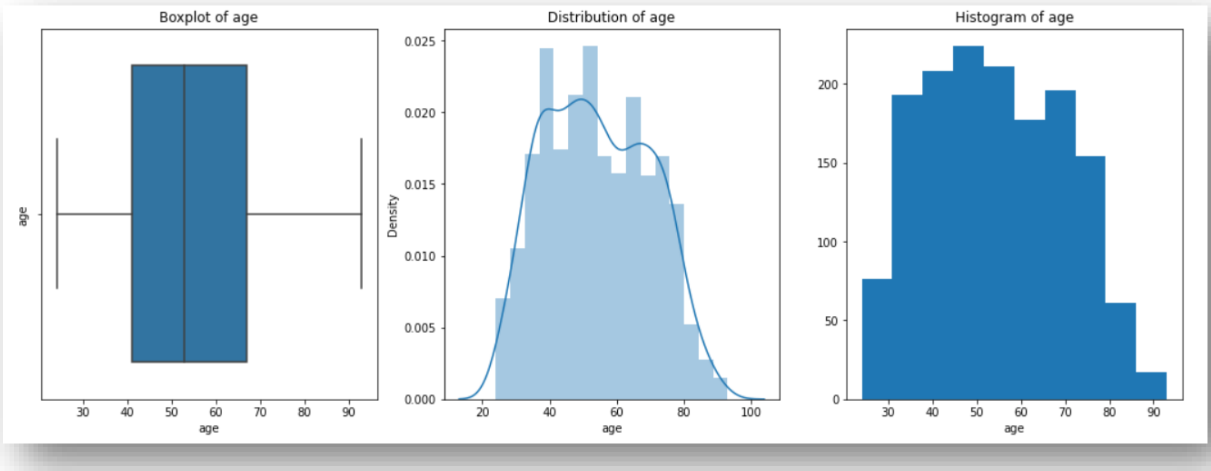
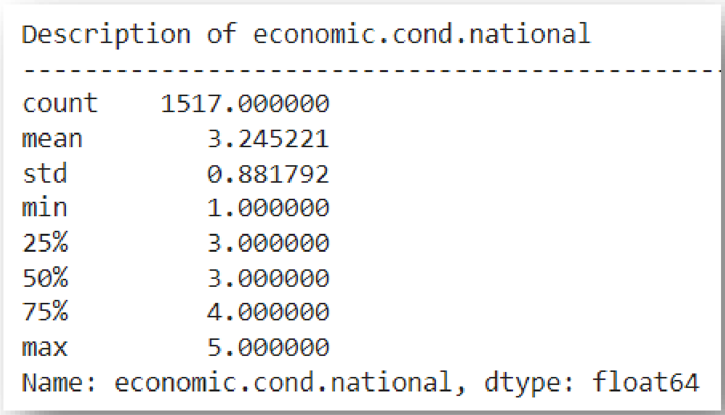


Figure 2: Boxplot, univariate distribution and histogram of 'age'

Univariate analysis of 'age' is done to understand the patterns and distribution of the data. From Figure 2, we can see that the Box plot of 'age' variable has no outliers. The distribution of the data is moderately right skewed which is seen in Figure 2. This is also seen in Table 10 where the skewness values are given. The skewness value of 'age' variable is 0.139800. From Table 11, it is seen that the mean of the data is 54.241266 meaning the age is 54 years on average.

2. economic.cond.national

Table 12: Description of 'economic.cond.national'



```
Description of economic.cond.national
------------------------------------------------
count    1517.000000
mean        3.245221
std         0.881792
min         1.000000
25%         3.000000
50%         3.000000
75%         4.000000
max         5.000000
Name: economic.cond.national, dtype: float64
```

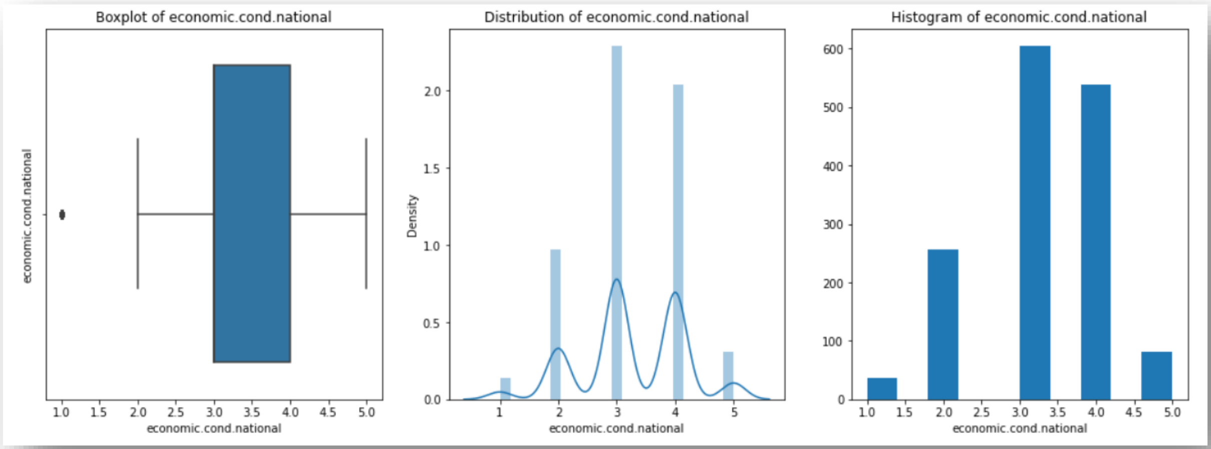
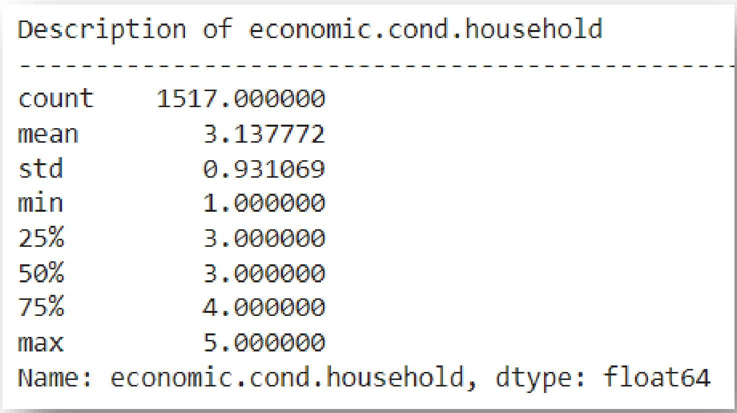


Figure 3: Boxplot, univariate distribution and histogram of 'economic.cond.national'

Univariate analysis of 'economic.cond.national' is done to understand the patterns and distribution of the data. From Figure 3, we can see that the Box plot of 'economic.cond.national' variable has an outlier. The distribution of the data is moderately left skewed which is seen in Figure 3. This is also seen in Table 10 where the skewness values are given. The skewness value of 'economic.cond.national' variable is -0.238474. From Table 12, it is seen that the mean of the data is 3.245221 meaning the assessment of current national economic conditions on a scale of 1 to 5 (1 being the worst and 5 being the best) is 3 on average.

3. economic.cond.household

Table 13: Description of 'economic.cond.household'



```
Description of economic.cond.household
----------------------------------------------
count    1517.000000
mean        3.137772
std         0.931069
min         1.000000
25%         3.000000
50%         3.000000
75%         4.000000
max         5.000000
Name: economic.cond.household, dtype: float64
```

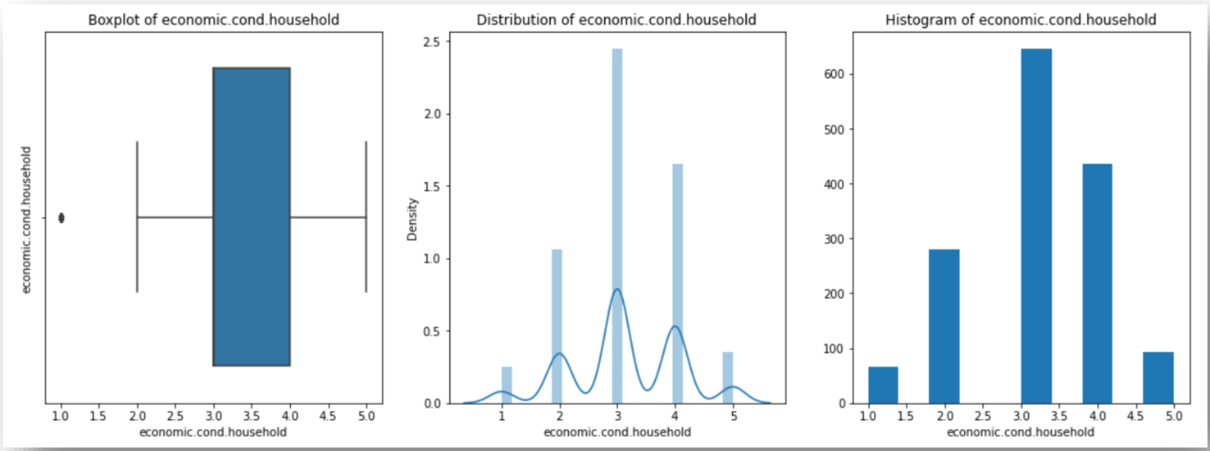
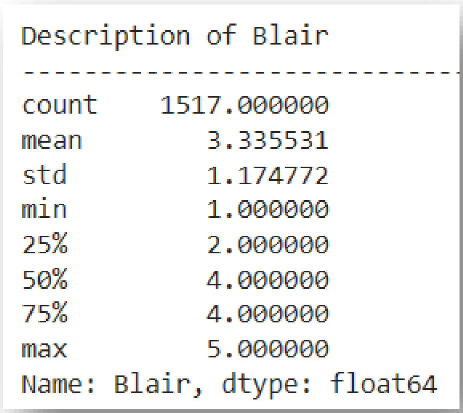


Figure 4: Boxplot, univariate distribution and histogram of 'economic.cond.household'

Univariate analysis of 'economic.cond.household' is done to understand the patterns and distribution of the data. From Figure 4, we can see that the Box plot of 'economic.cond.household' variable has an outlier. The distribution of the data is moderately left skewed which is seen in Figure 4. This is also seen in Table 10 where the skewness values are given. The skewness value of 'economic.cond.household' variable is -0.144148. From Table 13, it is seen that the mean of the data is 3.137772 meaning the assessment of current household economic conditions on a scale of 1 to 5 (1 being the worst and 5 being the best) is 3 on average.

4. Blair

Table 14: Description of 'Blair'



```
Description of Blair
------------------------------
count    1517.000000
mean        3.335531
std         1.174772
min         1.000000
25%         2.000000
50%         4.000000
75%         4.000000
max         5.000000
Name: Blair, dtype: float64
```

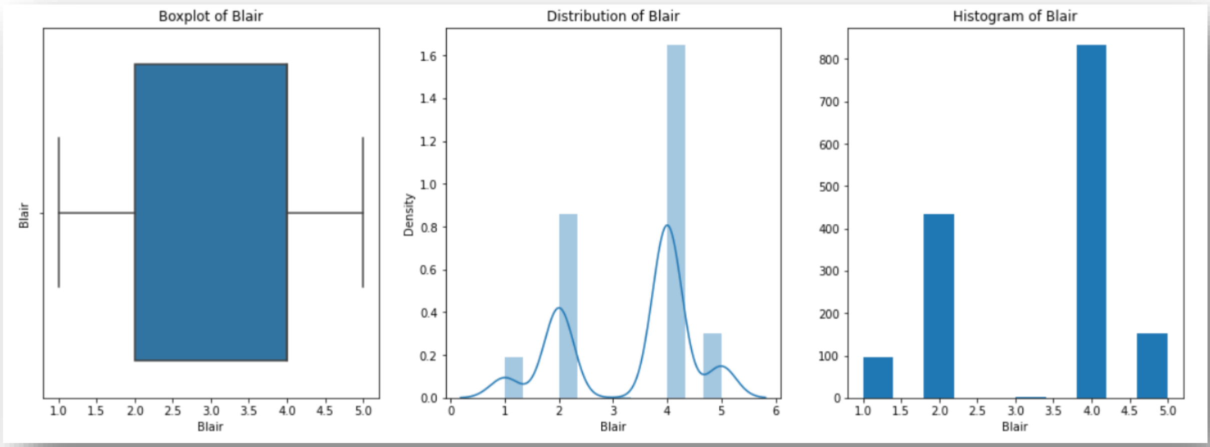
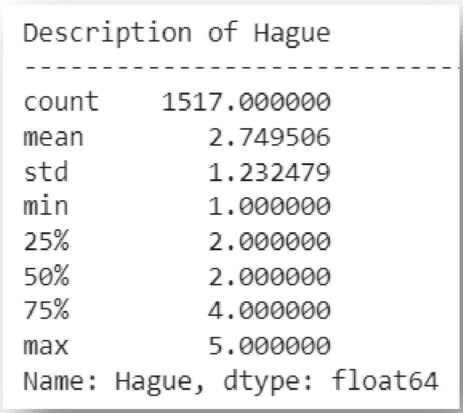


Figure 5: Boxplot, univariate distribution and histogram of 'Blair'

Univariate analysis of 'Blair' is done to understand the patterns and distribution of the data. From Figure 5, we can see that the Box plot of 'Blair' variable has no outliers. The distribution of the data is moderately left skewed which is seen in Figure 5. This is also seen in Table 10 where the skewness values are given. The skewness value of 'Blair' variable is -0.539514. From Table 14, it is seen that the mean of the data is 3.335531 meaning the assessment of the Labour leader on a scale of 1 to 5 (1 being the worst and 5 being the best) is 3 on average.

5. Hague

Table 15: Description of 'Hague'



```
Description of Hague
----------------------------
count    1517.000000
mean        2.749506
std         1.232479
min         1.000000
25%         2.000000
50%         2.000000
75%         4.000000
max         5.000000
Name: Hague, dtype: float64
```

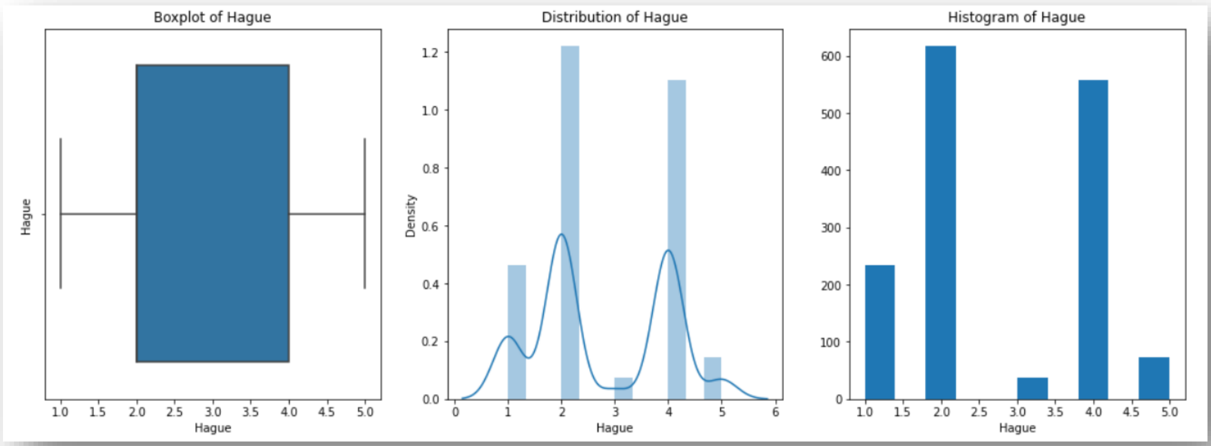
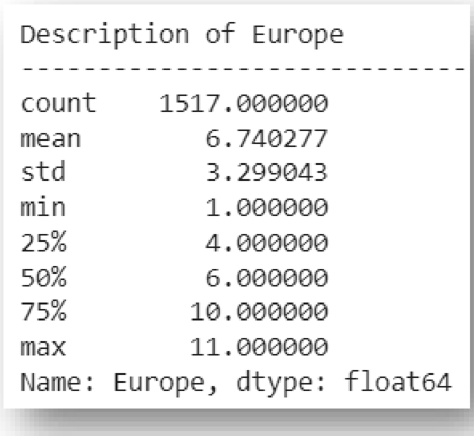


Figure 6: Boxplot, univariate distribution and histogram of 'Hague'

Univariate analysis of 'Hague' is done to understand the patterns and distribution of the data. From Figure 6, we can see that the Box plot of 'Hague' variable has no outliers. The distribution of the data is moderately right skewed which is seen in Figure 6. This is also seen in Table 10 where the skewness values are given. The skewness value of 'Hague' variable is 0.146191. From Table 15, it is seen that the mean of the data is 2.749506 meaning the assessment of the Conservative leader on a scale of 1 to 5 (1 being the worst and 5 being the best) is 3 on average.

6. Europe

Table 16: Description of 'Europe'



```
Description of Europe
-----------------------------
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
```

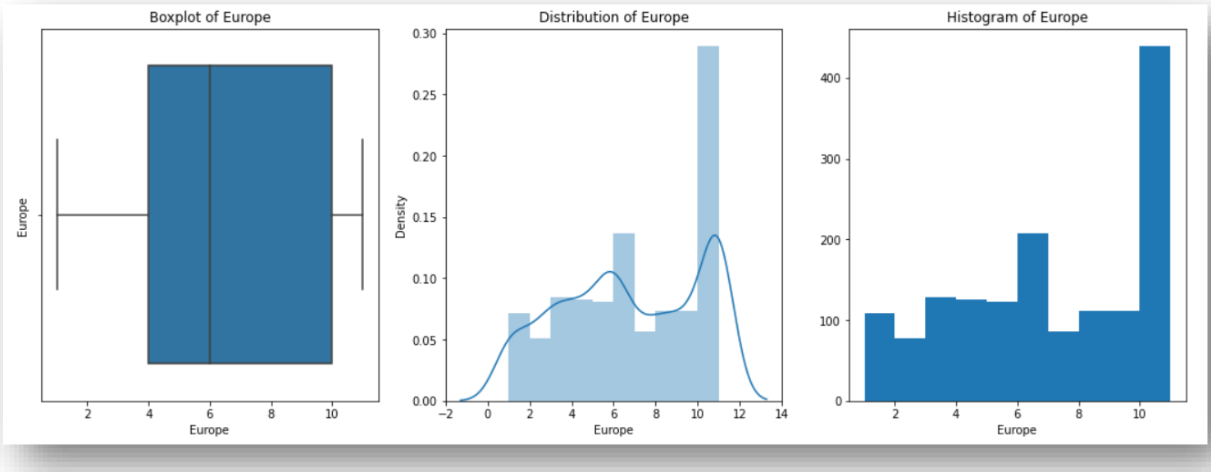
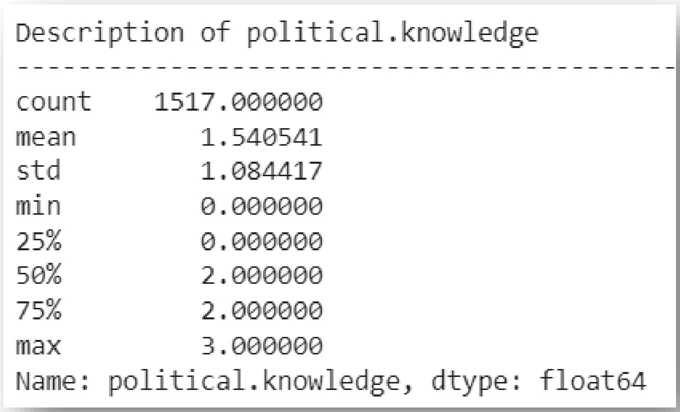


Figure 7: Boxplot, univariate distribution and histogram of 'Europe'

Univariate analysis of 'Europe' is done to understand the patterns and distribution of the data. From Figure 7, we can see that the Box plot of 'Europe' variable has no outliers. The distribution of the data is moderately left skewed which is seen in Figure 7. This is also seen in Table 10 where the skewness values are given. The skewness value of 'Europe' variable is -0.141891. From Table 16, it is seen that the mean of the data is 6.740277 meaning respondents' attitudes toward European integration is 6 on average on a scale of 1 to 11. High scores represent 'Eurosceptic' sentiment.

7. Political.knowledge

Table 17: Description of 'political.knowledge'



```
Description of political.knowledge
-------------------------------------------
count    1517.000000
mean        1.540541
std         1.084417
min         0.000000
25%         0.000000
50%         2.000000
75%         2.000000
max         3.000000
Name: political.knowledge, dtype: float64
```

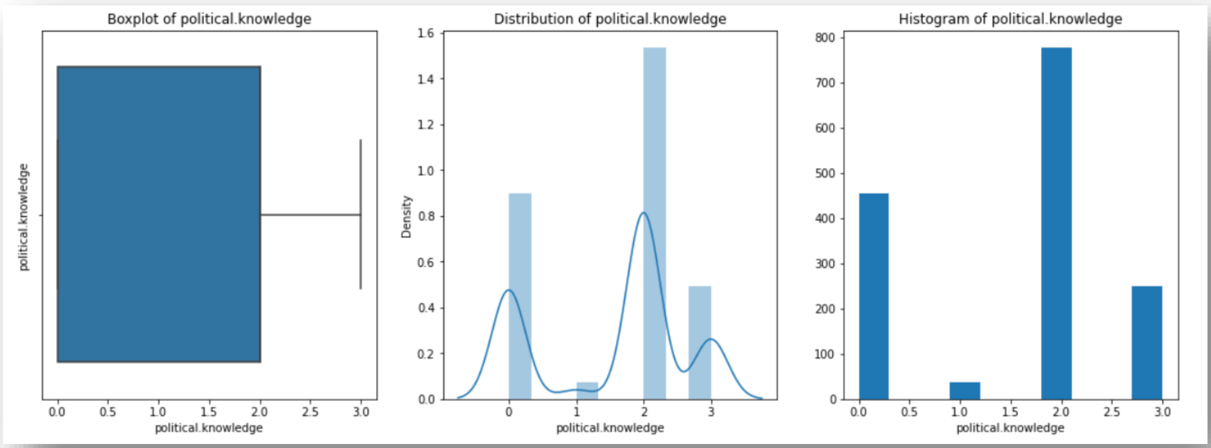


Figure 8: Boxplot, univariate distribution and histogram of 'political.knowledge'

Univariate analysis of 'political.knowledge' is done to understand the patterns and distribution of the data. From Figure 8, we can see that the Box plot of 'political.knowledge' variable has no outliers. The distribution of the data is moderately left skewed which is seen in Figure 8. This is also seen in Table 10 where the skewness values are given. The skewness value of 'political.knowledge' variable is -0.422928. From Table 17, it is seen that the mean of the data is 1.540541 meaning the knowledge of parties' positions on European integration is 1 on average on a scale of 0 to 3.

Categorical variables:

8. Vote

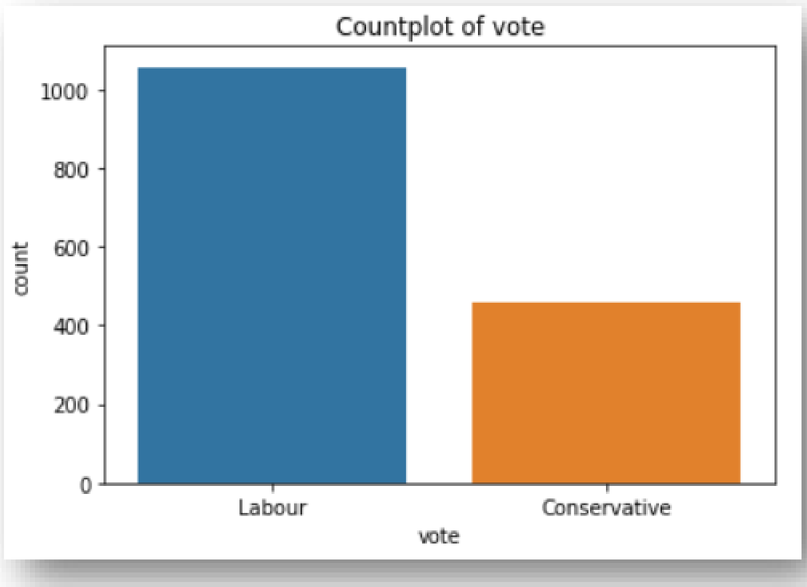


Figure 9: Countplot of 'vote'

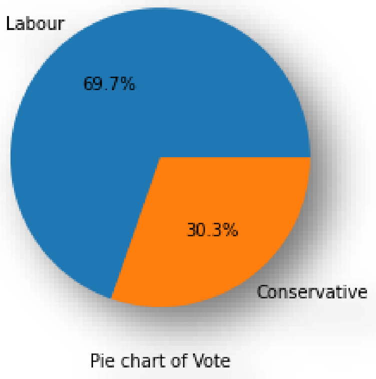


Figure 10: Pie chart of 'vote'

Univariate analysis of ‘vote’ is done to understand the patterns and distribution of the data. From Figure 9 and Figure 10, we can understand that ‘Labour’ is the most voted choice of Party and ‘Conservative’ is the least voted choice of Party.

9. Gender

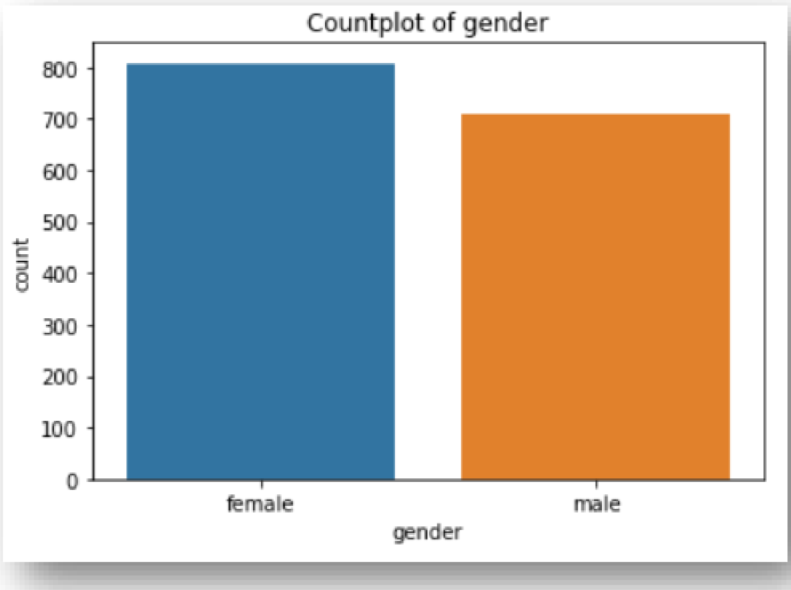


Figure 11: Countplot of 'gender'

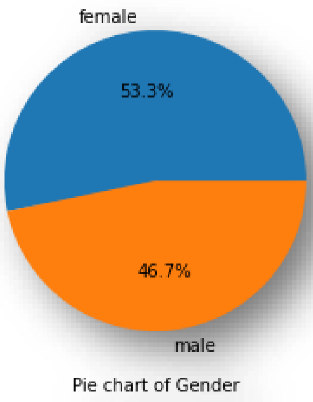
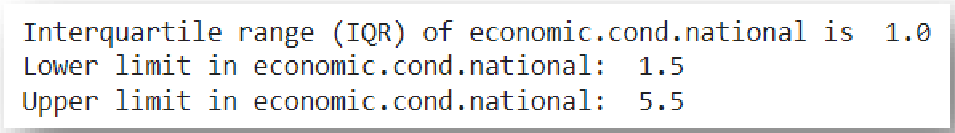


Figure 12: Pie chart of 'gender'

Univariate analysis of ‘gender’ is done to understand the patterns and distribution of the data. From Figure 11 and Figure 12, we can understand that majority of the votes is cast by ‘female’.

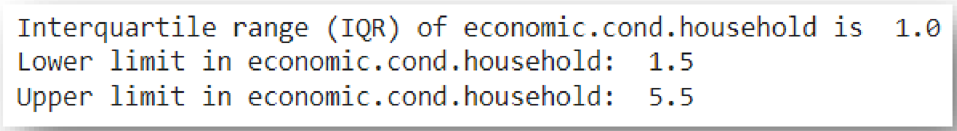
Outliers:

There are 2 variables with outliers present. They are ‘economic.cond.national’ and ‘economic.cond.household’ which can be seen from the boxplots in Figure 3 and Figure 4. The upper and lower limits is found to get a clear picture of the outliers.



```
Interquartile range (IQR) of economic.cond.national is  1.0
Lower limit in economic.cond.national:  1.5
Upper limit in economic.cond.national:  5.5
```

Figure 13: Upper and lower limit of 'economic.cond.national'



```
Interquartile range (IQR) of economic.cond.household is  1.0
Lower limit in economic.cond.household:  1.5
Upper limit in economic.cond.household:  5.5
```

Figure 14: Upper and lower limit of 'economic.cond.household'

The upper and lower limits are shown in Figure 13 and Figure 14. They are not distant from each other and the outliers are on the lower side having the value 1 while the lower limit is 1.5. Since the difference between the outlier and the lower limit is very less, the outliers are not treated in this case.

Bivariate Analysis:

Numerical variables:

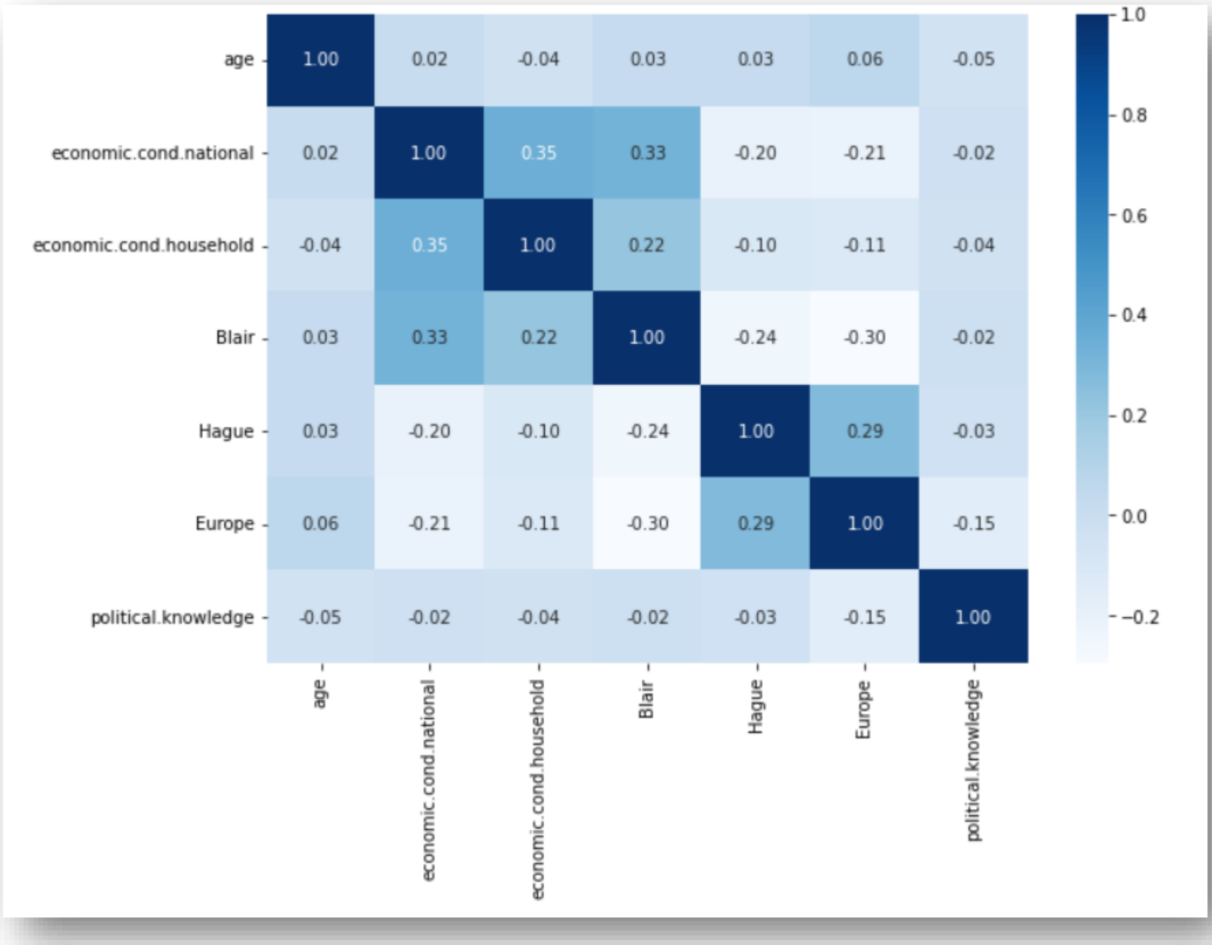


Figure 15: Heat map showing the bivariate analysis of the dataset

Bivariate analysis is done using the help of a heat map. A heat map is used to understand the correlation between two numerical values in a dataset. Multicollinearity is an important issue which can destruct the model. Heatmap can help in identifying this issue. Figure 15 shows the heatmap of the dataset.

Observations:

- The highest correlation is between the different features like 'economic_cond_national' and 'economic_cond_household' (35%) although it is not a strong correlation
- There is less correlation in table with the other features
- Europe is most negatively correlated with Blair (-30%)
- Therefore, multicollinearity will not be an issue in this dataset

1. Gender

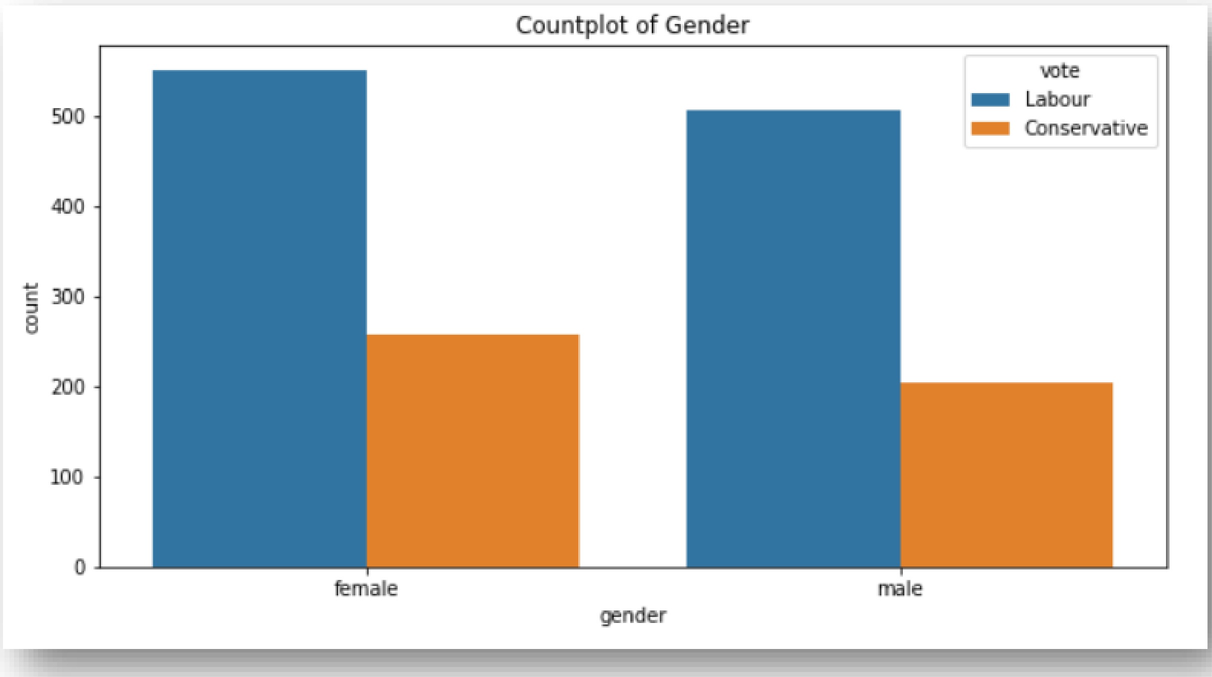


Figure 16: Countplot of 'gender' and 'vote'

From Figure 16, it is seen that the majority of the votes has been casted for 'Labour' party regardless of the gender and among the gender, 'female' has casted the highest number of votes.

2. Economic.cond.national

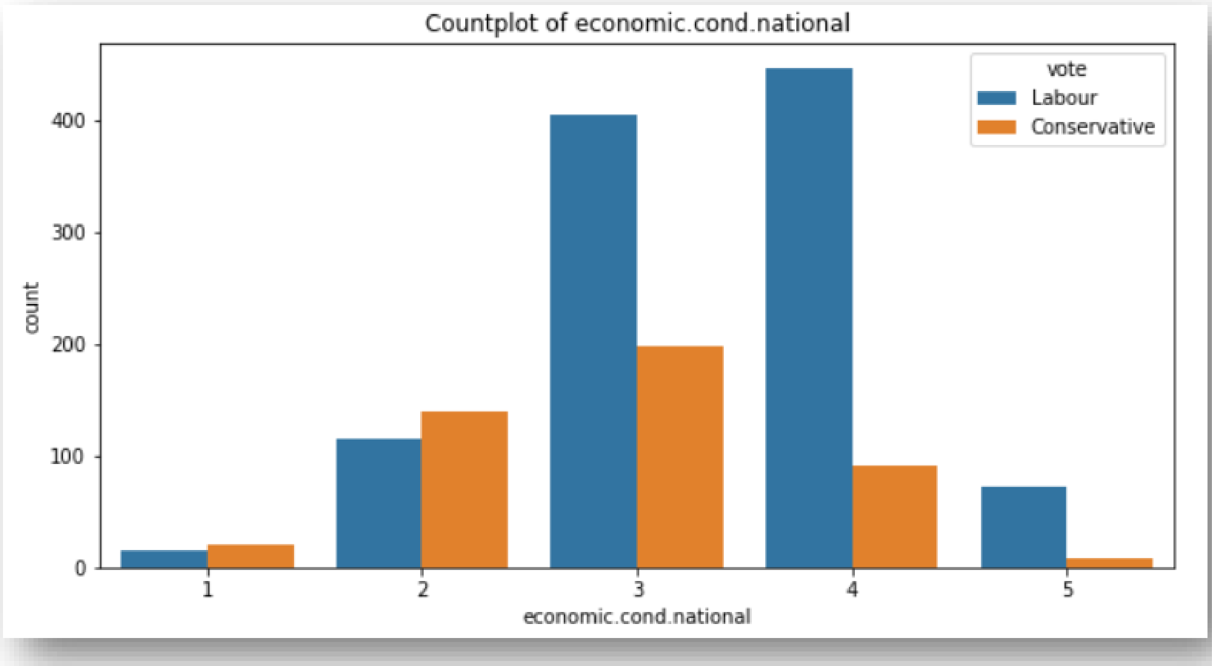


Figure 17: Countplot of 'economic.cond.national' and 'vote'

From Figure 17, it is seen that people with better national economic conditions are preferring to vote for ‘Labour’ Party.

3. Economic.cond.household

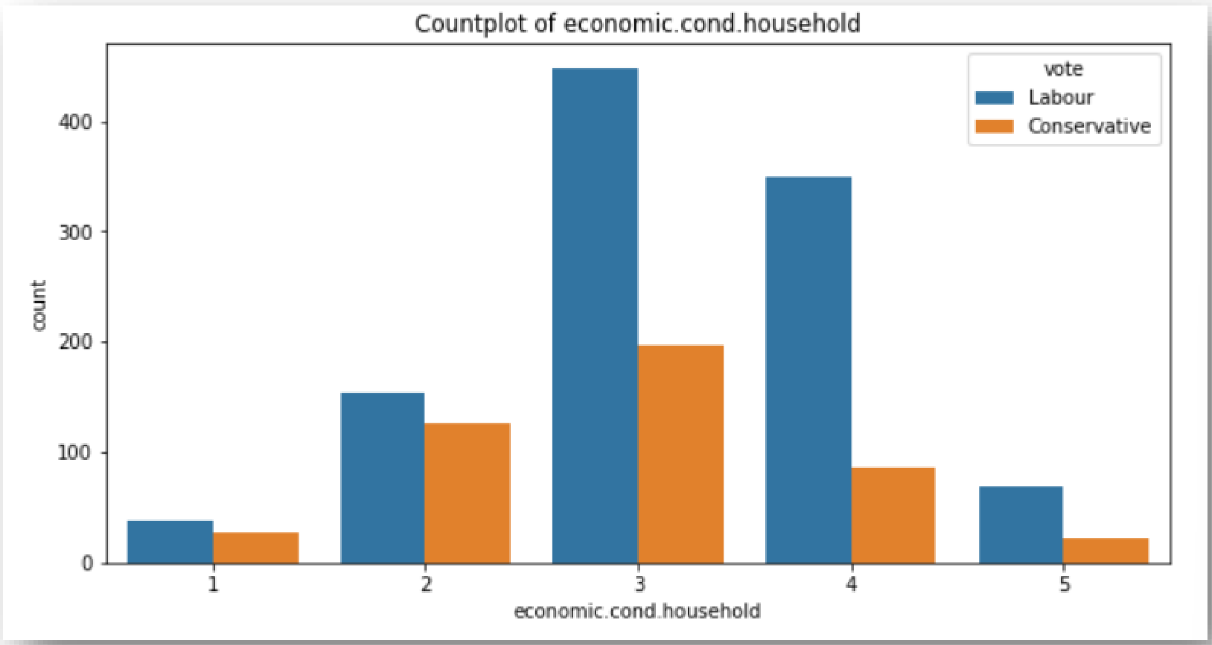


Figure 18: Countplot of 'economic.cond.household' and 'vote'

From Figure 18, it is seen that people with better household economic conditions are preferring to vote for ‘Labour’ Party.

4. Blair

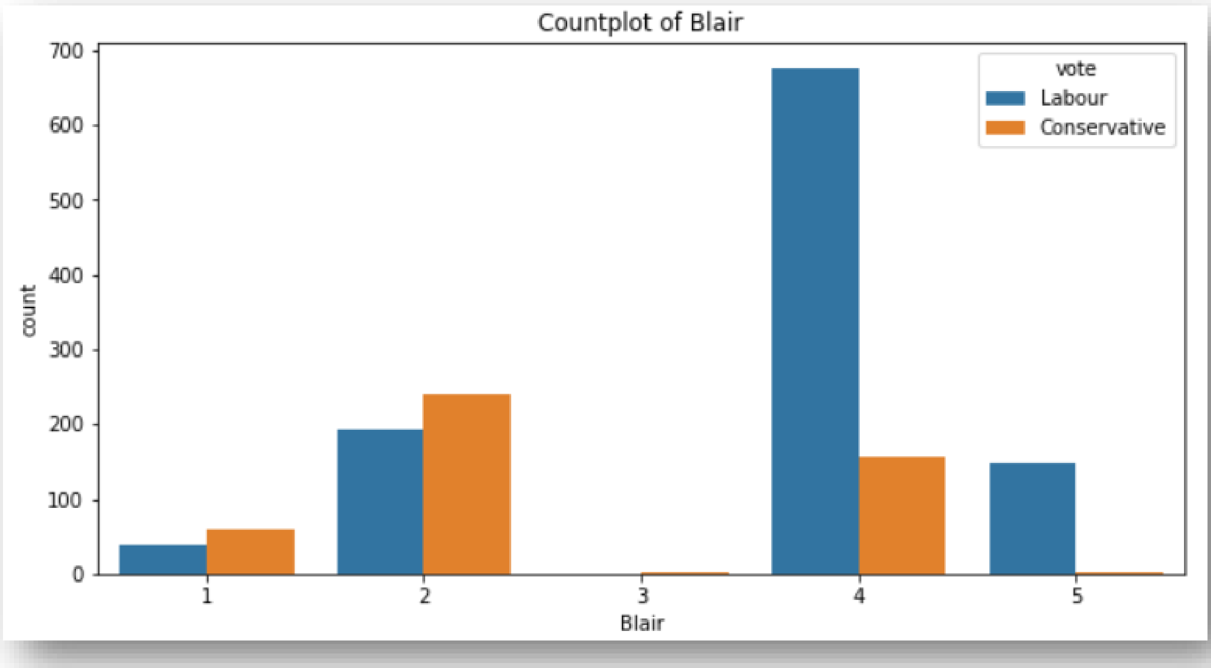


Figure 19: Countplot of 'Blair' and 'vote'

From Figure 19, it is seen that the assessment of the labour leader is good with a score of 4 as the majority.

5. Hague

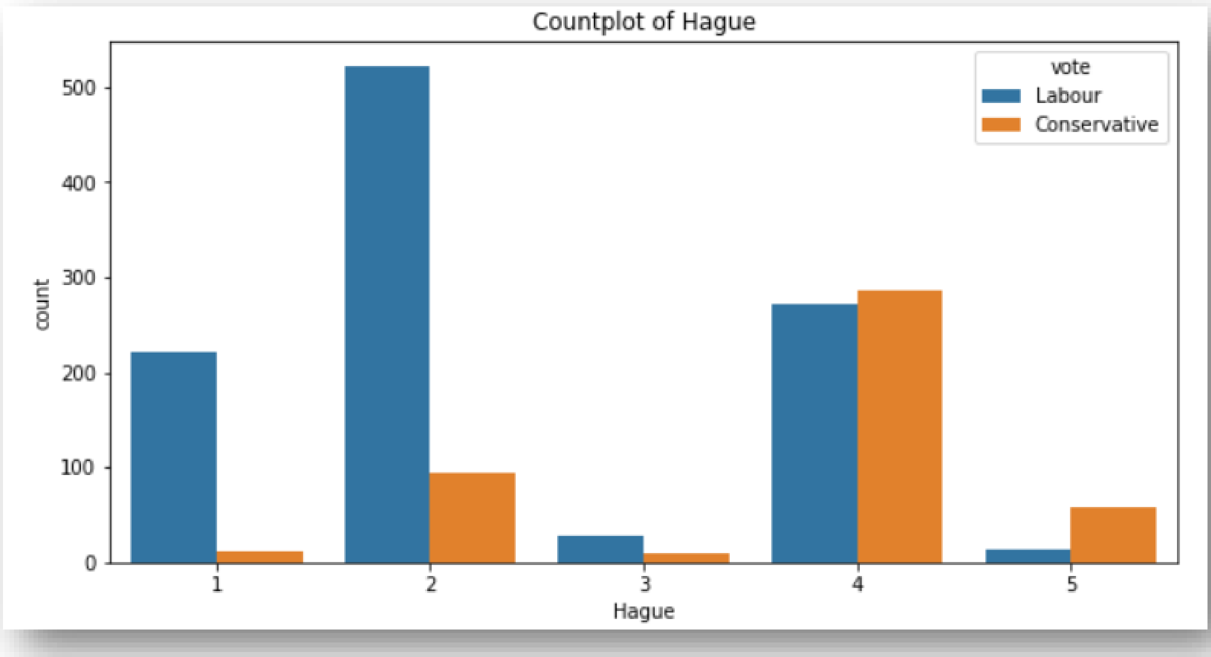


Figure 20: Countplot of 'Hague' and 'vote'

From Figure 20, it is seen that the assessment of the conservative leader is good as the majority score is 4. However, the scores for the labour leader were better.

6. Europe

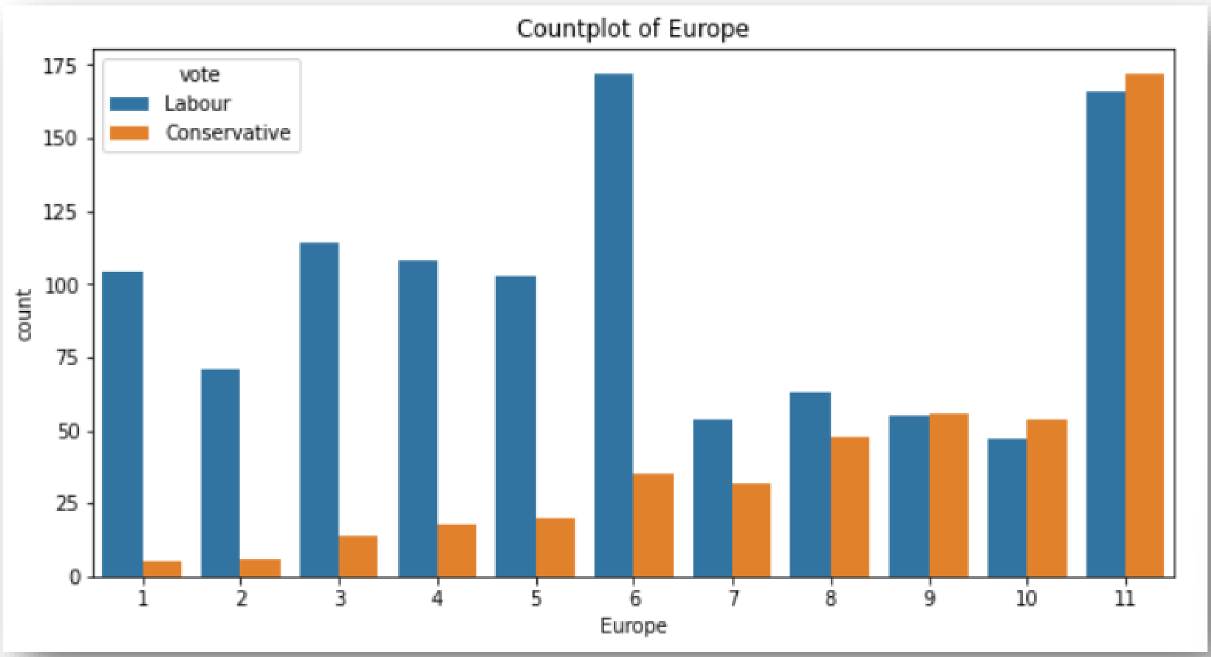


Figure 21: Countplot of 'Europe' and 'vote'

From Figure 21, it is seen that people with high 'Eurosceptic' sentiment are preferring to vote for conservative party.

7. Political.knowledge

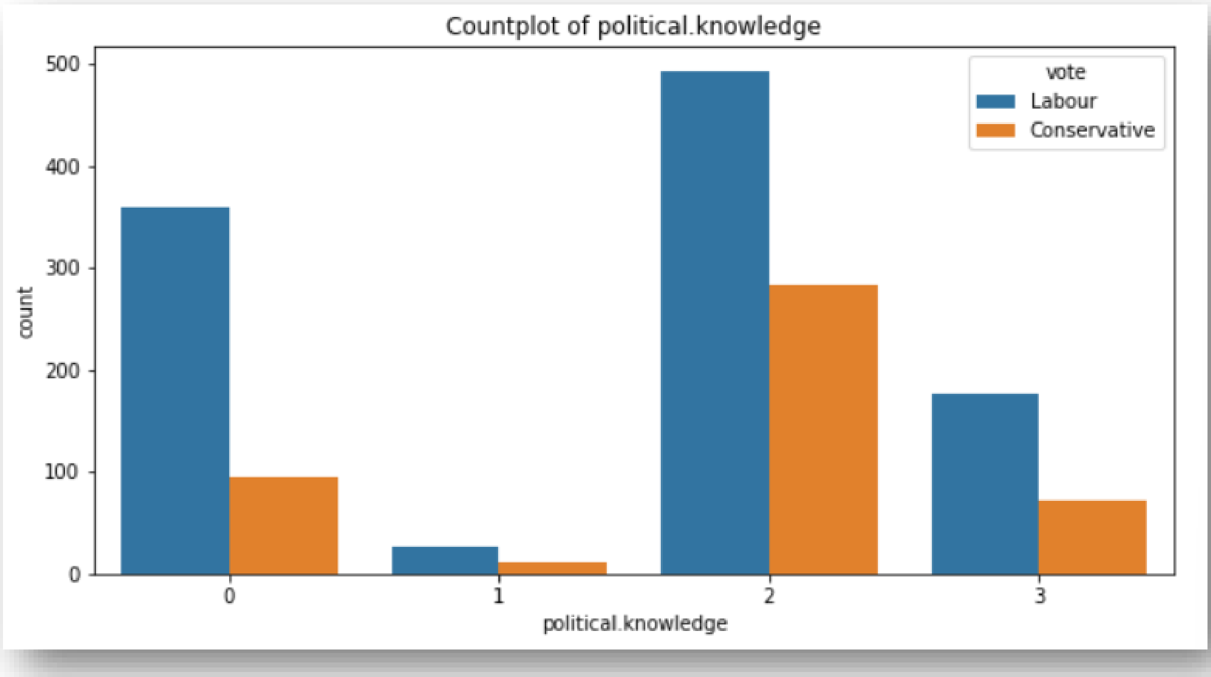


Figure 22: Countplot of 'political.knowledge' and 'vote'

From Figure 22, it is seen that people who have high political knowledge have voted for conservative party.

Multivariate Analysis:

Numerical variables:

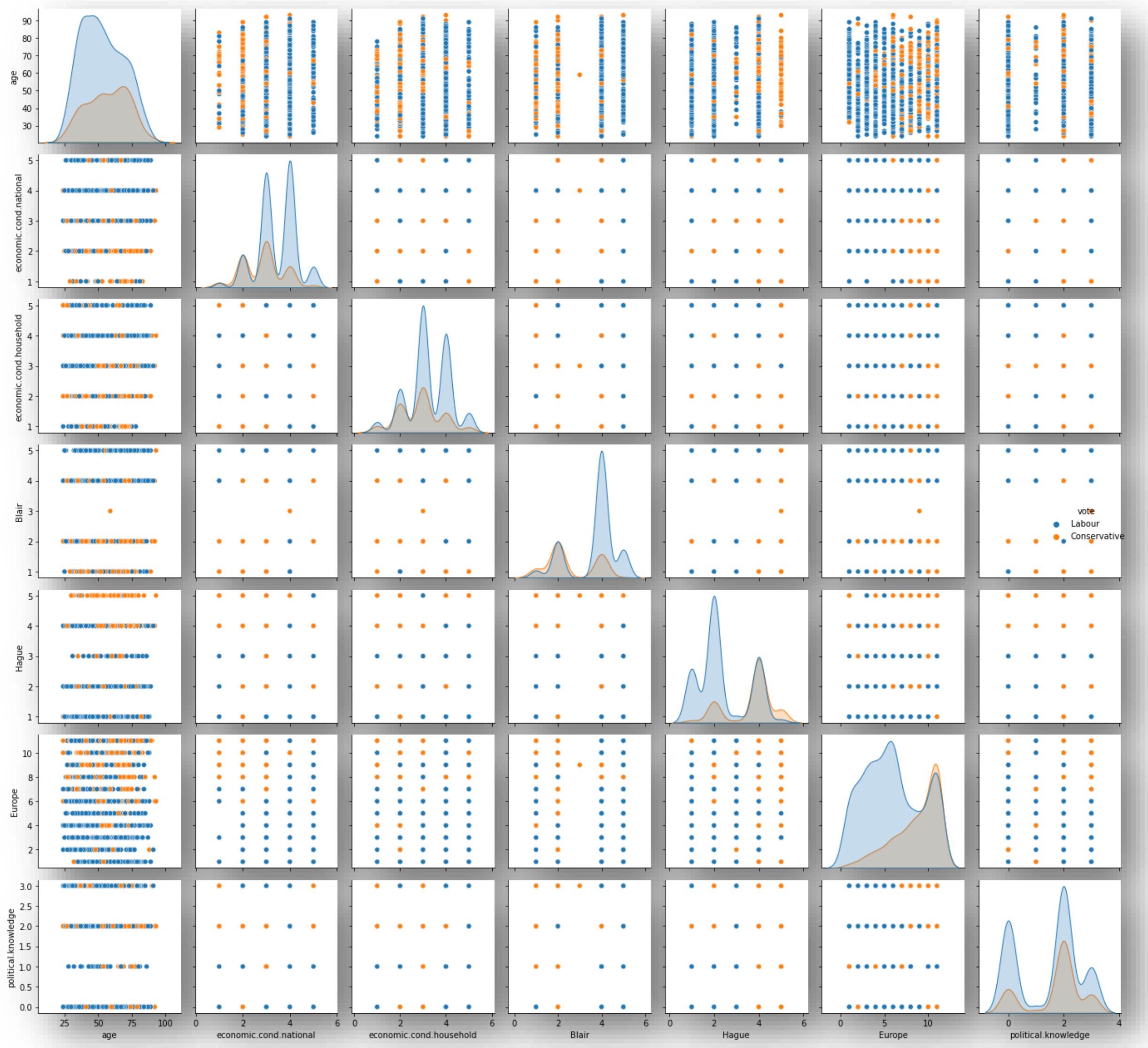


Figure 23: Pair plot showing the multivariate analysis of the numerical variables in the dataset

Multivariate analysis is done using the help of a pair plot to understand the relationship between all the numerical values in the dataset. Pair plot can be used to compare all the variables with each other to understand the patterns or trends in the dataset. Figure 23 shows the pair plot of the dataset.

Observations:

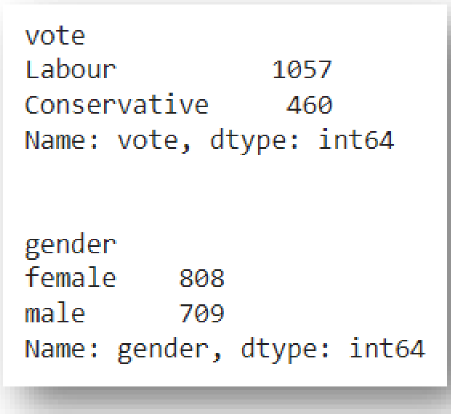
- There is no strong relationship present between the variables
- There is a mixture of positive and negative relationships

Data Preparation: 4 marks

1.3 Encode the data (having string values) for Modelling. Is Scaling necessary here or not? Data Split: Split the data into train and test (70:30). (4 Marks)

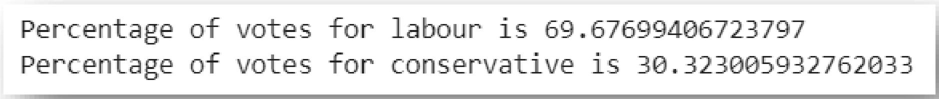
Encoding the dataset:

Table 18: Value counts of the categorical variables in the dataset



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


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

There are 2 categorical variables in this dataset – 'vote' and 'gender'. Table 18 shows the different categorical variables and the value counts for each of the types in the different categories.



```
Percentage of votes for labour is 69.67699406723797
Percentage of votes for conservative is 30.323005932762033
```

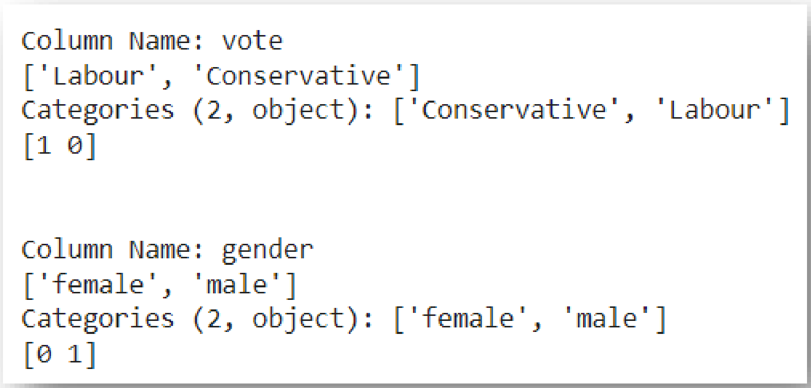
Figure 24: Split of the dependent variable data

From Figure 24 it is seen that the data split is not equal. 69.7% of the votes are for labour party and 30.3% of the votes is for conservative party.

Many models in machine learning require all the features to be numerical. The models cannot work with string values. Therefore, the categorical variables need to be encoded in order to convert the datatype to integer type.

There are 2 categorical variables – 'vote' and 'gender' in the dataset shown in Table 18. Therefore, these variables must be encoded.

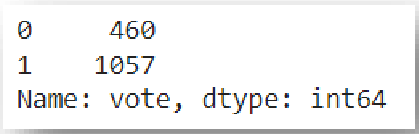
Table 19: Categorical variables after encoding



```
Column Name: vote
['Labour', 'Conservative']
Categories (2, object): ['Conservative', 'Labour']
[1 0]


Column Name: gender
['female', 'male']
Categories (2, object): ['female', 'male']
[0 1]
```

Table 18 shows the distribution of the 2 categorical columns of the dataset. It is seen that both the variables have two classifications of data in them. They are converted using categorical conversion methods. This will convert the 2 classifications of data values into 0 and 1. There is no level or order in the subcategory. Therefore, any method of encoding can be used and the result will be the same. Table 19 shows the classification of the data in each of the categorical variables and the results after encoding.

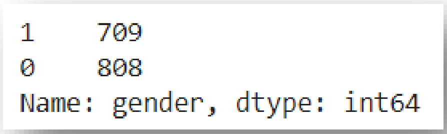
Table 20: Value conuts of 'vote' variable



```
0     460
1    1057
Name: vote, dtype: int64
```

From Table 20, it is seen that after encoding, the value counts are 460 for conservative and 1057 for labour.

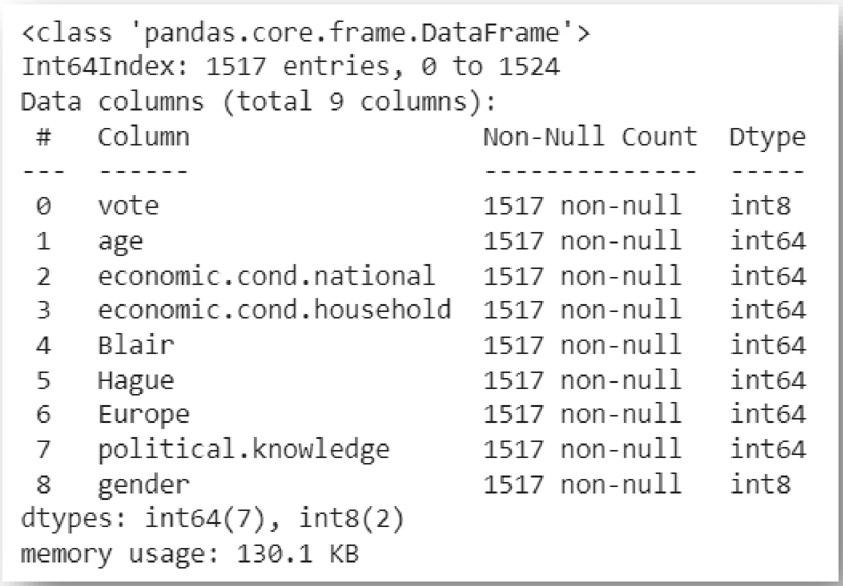
Table 21: Value counts of 'gender' variable



```
1    709
0    808
Name: gender, dtype: int64
```

From Table 21, it is seen that after encoding, the value counts are 709 for male and 808 for female.

Table 22: Information of the dataset after encoding



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1517 entries, 0 to 1524
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   vote                     1517 non-null   int8
 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   int8
dtypes: int64(7), int8(2)
memory usage: 130.1 KB
```

From Table 22, it is seen that all the columns are of integer data type. Therefore, the dataset can now be used for building models.

Scaling:

Scaling is a way of representing a dataset. Scaling needs to be done as a dataset has features or variables with different 'weights' for each feature. The independent attributes have different units and scales of measurement. In such cases, it is suggested to transform the features so that all the features are in the same 'scale'. This is called scaling. Scaling also makes it easier to compare the features now that the weightage on each feature is the same. Therefore, it will bring out the best performance of the model.

Depending on the model, we can decide if scaling is required or not. Scaling can be done by Z-score method and Min-max method. Z-score method is used when you want to centralize the data (weight-based). Example: principal component analysis (PCA), neural network etc., Min-max method is used when the data is distance based. Example: Clustering, KNN etc.,

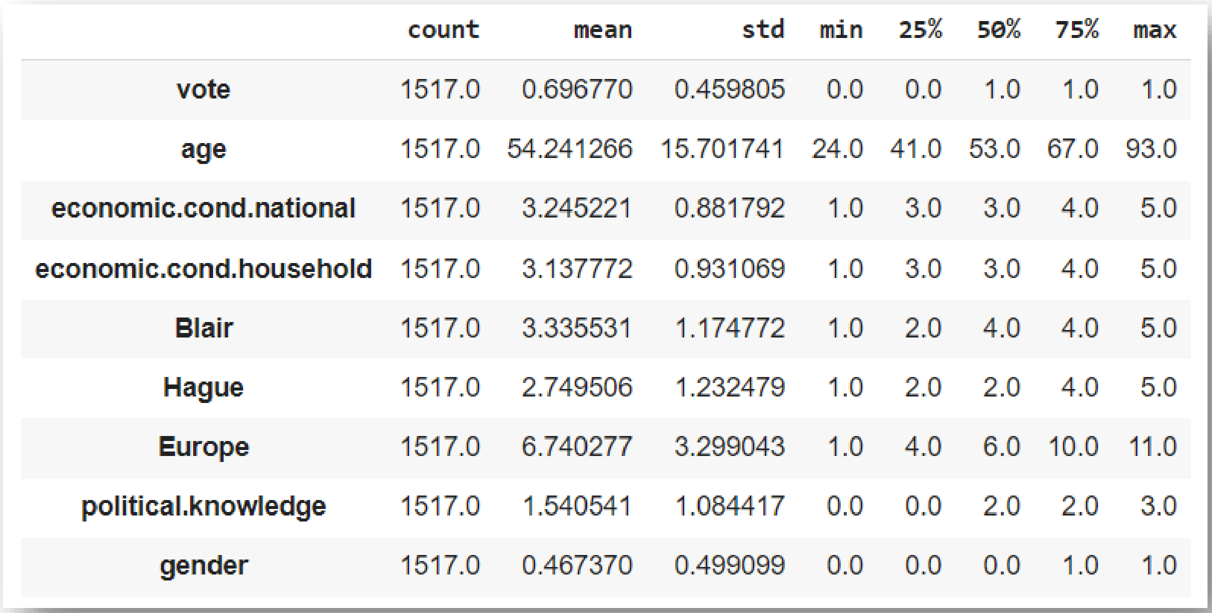
Gradient descent algorithms like linear regression and logistic regression are sensitive to ranges of data points. Distance-based models like KNN also require scaling as it is sensitive to extreme values which will affect the model whereas tree-based models such as decision trees would not require scaling as it uses split method (hence, it is unnecessary). For these reasons, the **scaled data** will be used to build **logistic regression model and KNN model**. Models can be implemented after scaling is done.

Scaling is done only for the continuous variables. In this case only 'age' is a continuous variable while the other variables are categorical variables which are coded. Therefore Min-max method is used for scaling the dataset in this case as the binary values will not change. The scaled data will range between 0 and 1.

$$Min - \max = \frac{(X - Xmin)}{(Xmax - Xmin)}$$

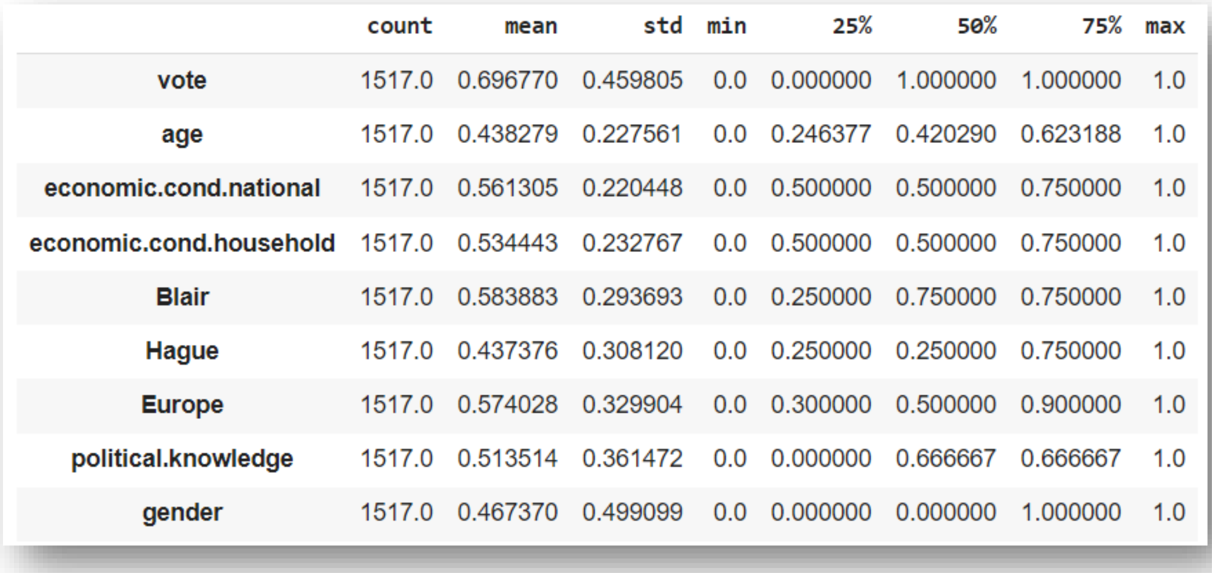
0 will remain >> (0-0)/(1-0) = 0
1 will remain >> (1-0)/(1-0) = 1

Table 23: Description of the dataset before scaling



	count	mean	std	min	25%	50%	75%	max
vote	1517.0	0.696770	0.459805	0.0	0.0	1.0	1.0	1.0
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
gender	1517.0	0.467370	0.499099	0.0	0.0	0.0	1.0	1.0

Table 24: Description of the dataset after scaling



	count	mean	std	min	25%	50%	75%	max
vote	1517.0	0.696770	0.459805	0.0	0.000000	1.000000	1.000000	1.0
age	1517.0	0.438279	0.227561	0.0	0.246377	0.420290	0.623188	1.0
economic.cond.national	1517.0	0.561305	0.220448	0.0	0.500000	0.500000	0.750000	1.0
economic.cond.household	1517.0	0.534443	0.232767	0.0	0.500000	0.500000	0.750000	1.0
Blair	1517.0	0.583883	0.293693	0.0	0.250000	0.750000	0.750000	1.0
Hague	1517.0	0.437376	0.308120	0.0	0.250000	0.250000	0.750000	1.0
Europe	1517.0	0.574028	0.329904	0.0	0.300000	0.500000	0.900000	1.0
political.knowledge	1517.0	0.513514	0.361472	0.0	0.000000	0.666667	0.666667	1.0
gender	1517.0	0.467370	0.499099	0.0	0.000000	0.000000	1.000000	1.0

From Table 23 and Table 24, we are able to see how the values have changed in scale. The values may seem to be different but however they are only scaled. The dataset is brought into one unit of comparison. To prove this, a histogram is plotted.

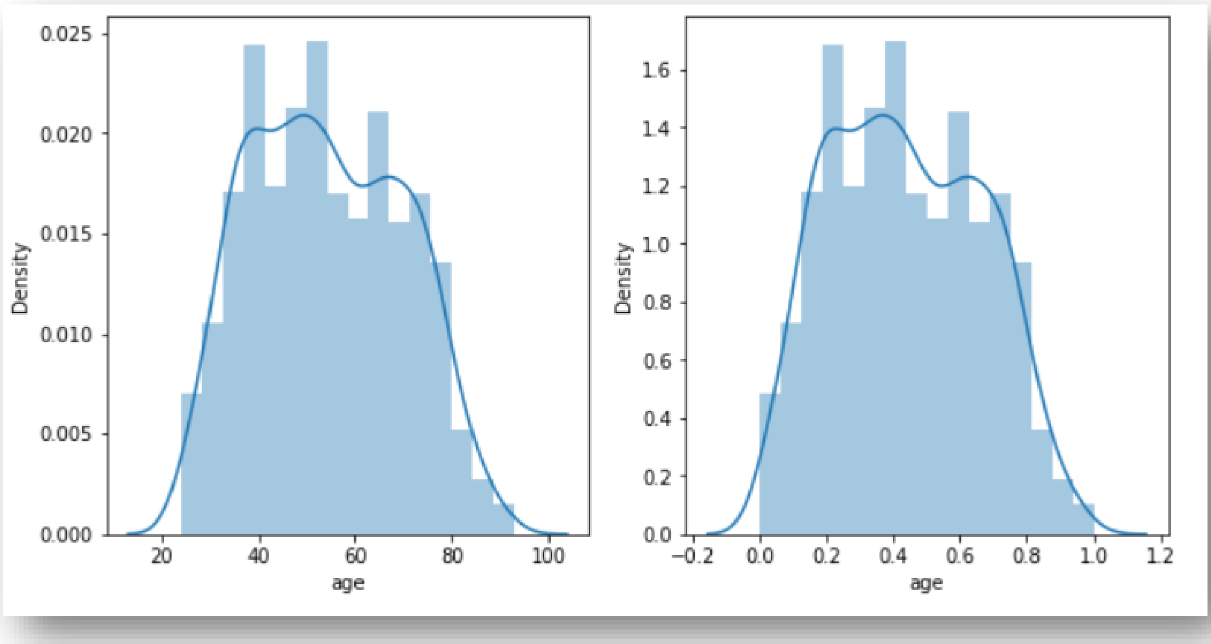


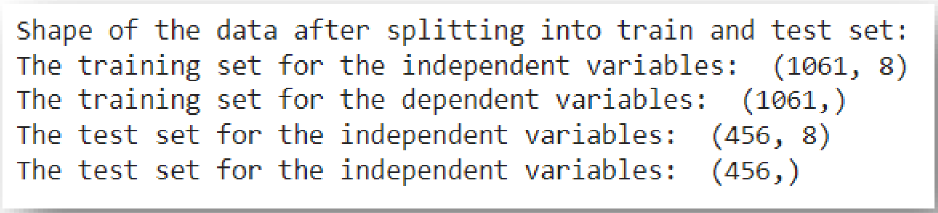
Figure 25: Graph showing the histogram of 'age' variable before scaling (left) and the histogram of 'age' variable after scaling by Min-max method(right)

From Figure 25, we can see that the plot of the 'age' variable before and after scaling is the same but the scale has changed. This is the case for all the other numerical variables that has been scaled using the min-max method.

Data split: Splitting the data into train and test

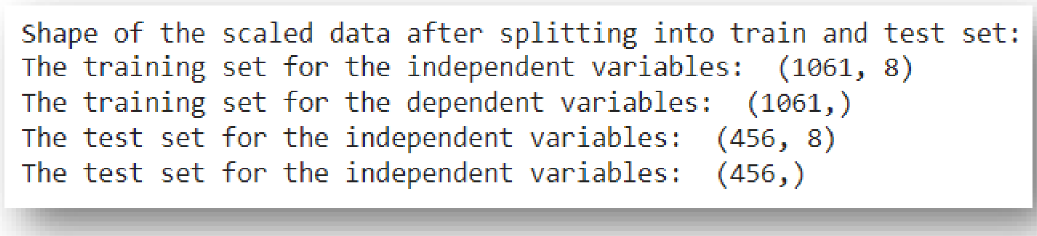
We will use two different train and test sets – one with scaled data and one without scaled data for building different models.

The data is first split into train and test data before building the models. There is no fixed rule for separation training and testing data sets. Most of the researchers use **70:30 ratio** for separation data sets. The same ratio was used in this dataset to split the data into train and test. The random state was set to be 1.



```
Shape of the data after splitting into train and test set:
The training set for the independent variables:  (1061, 8)
The training set for the dependent variables:  (1061,)
The test set for the independent variables:  (456, 8)
The test set for the independent variables:  (456,)
```

Figure 26: Shape of the train and test set without scaling the data



```
Shape of the scaled data after splitting into train and test set:
The training set for the independent variables:  (1061, 8)
The training set for the dependent variables:  (1061,)
The test set for the independent variables:  (456, 8)
The test set for the independent variables:  (456,)
```

Figure 27: Shape of the train and test set after scaling the data

The training set for the independent variables (X_train) denotes 70% training dataset with 8 columns (without the dependent variable – 'vote'). The training set for the dependent variable (y_train) denotes 70% test dataset with only the target column or the dependent variable 'vote'. The test set for the independent variables (X_test) denotes 30% training dataset with 8 columns (without the dependent variable – 'vote'). The test set for the dependent variable (y_test) denotes 30% test dataset with only the target column or the dependent variable 'vote'.

Figure 26 shows the shape of the train and test set without scaling the dataset and Figure 27 shows the shape of the train and test set after scaling the dataset.

Modeling: 22 marks

1.4 Apply Logistic Regression and LDA (linear discriminant analysis). (4 marks)

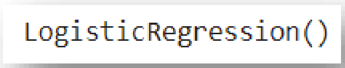
Logistic Regression

Logistic Regression model is supervised learning algorithm which can be used for classification type of problems. It establishes relation between dependent class variable and independent variables using regression.

The models are built with default parameters and later Grid Search CV is done with different parameters to see which parameters suits the best for our data.

The model is built with the default parameters automatically using LogisticRegression() function.

Scaled data

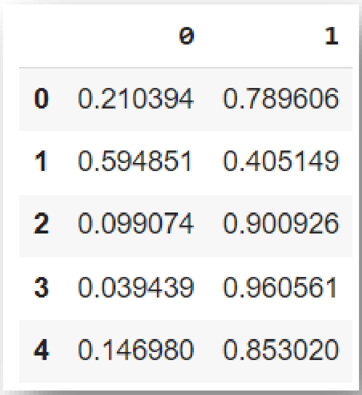


```
LogisticRegression()
```

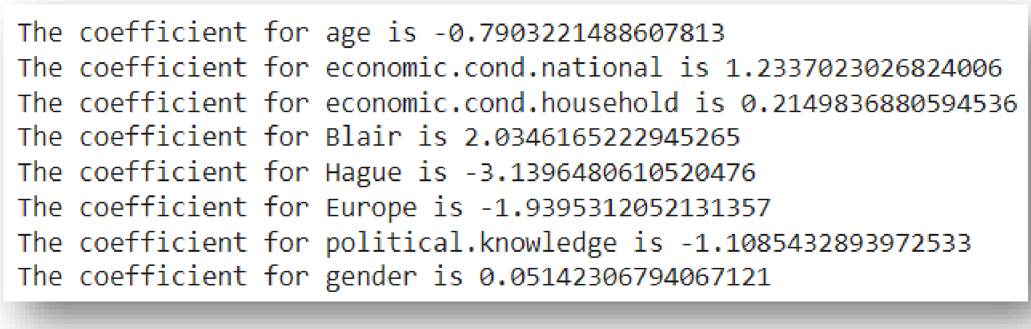
Figure 28: Logistic regression model built with default parameters

- The **'solver'** is the algorithm used in the process of backpropagation to calculate the weights of the coefficients in the logistic regression model. The default 'solver' is **'lbfgs'** solver.
- Penalized logistic regression imposes a penalty to the logistic model for having too many variables. The default **'penalty'** is **'l2'.**
- The **'max_iter'** parameter is the maximum number of iterations to update the synaptic weights of neurons. The default 'max_iter' value is **100**.

Table 25: Logistic regression - Probabilities on the test set



	0	1
0	0.210394	0.789606
1	0.594851	0.405149
2	0.099074	0.900926
3	0.039439	0.960561
4	0.146980	0.853020



```
The coefficient for age is -0.7903221488607813
The coefficient for economic.cond.national is 1.2337023026824006
The coefficient for economic.cond.household is 0.2149836880594536
The coefficient for Blair is 2.0346165222945265
The coefficient for Hague is -3.1396480610520476
The coefficient for Europe is -1.9395312052131357
The coefficient for political.knowledge is -1.1085432893972533
The coefficient for gender is 0.05142306794067121
```

Figure 29: Logistic regression - Coefficient values for each column

There are 8 attributes in this dataset and hence there are 8 coefficients. For every one-unit change in x (the different independent variables), y (the dependent variable) changes m (the coefficient value) times. The coefficient values for each column are shown in Figure 29.

The training accuracy of the model is 0.8265786993402451.
The test accuracy of the model is 0.8508771929824561.

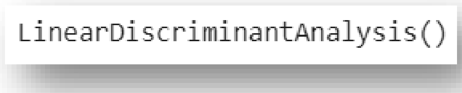
Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) uses linear combinations of independent variables to predict the class in the response variable of a given observation. LDA assumes that the independent variables (p) are normally distributed and there is equal variance / covariance for the classes. LDA is popular, because it can be used for both classification and dimensionality reduction.

The models are built with default parameters and later Grid Search CV is done with different parameters to see which parameters suits the best for our data.

The model was built with the default parameters automatically using LinearDiscriminantAnalysis() function.

Without scaling the data

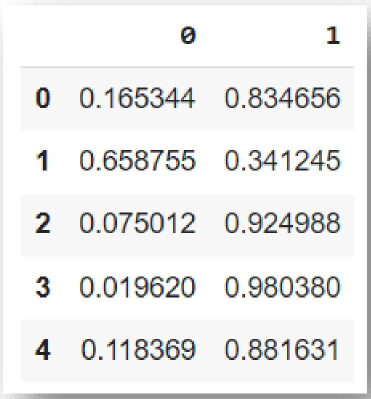


```
LinearDiscriminantAnalysis()
```

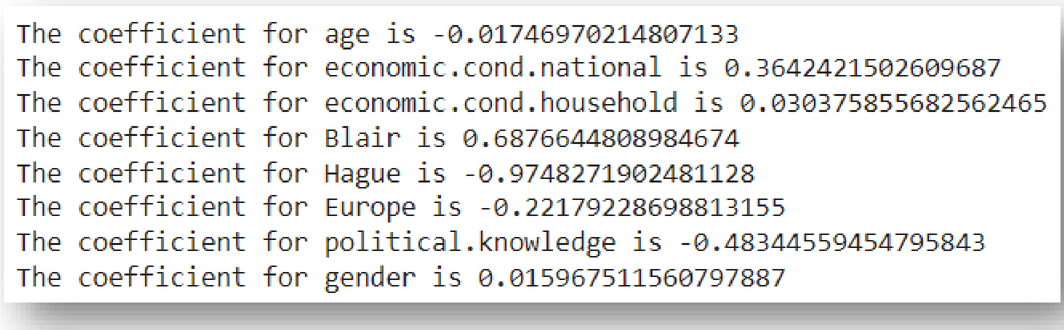
Figure 30: LDA model built with default parameters

- The **'solver'** is the algorithm used in the process of backpropagation to calculate the weights of the coefficients in the logistic regression model. The default 'solver' was found to be **'svd'** solver.
- The **'tol'** parameter is the threshold level. The lower the threshold, higher the accuracy and lesser the number of times the model will execute and vice versa. The default threshold value was found to be **'0.0001'**.

Table 26: LDA - Probabilities on the test set



	0	1
0	0.165344	0.834656
1	0.658755	0.341245
2	0.075012	0.924988
3	0.019620	0.980380
4	0.118369	0.881631



```
The coefficient for age is -0.01746970214807133
The coefficient for economic.cond.national is 0.3642421502609687
The coefficient for economic.cond.household is 0.030375855682562465
The coefficient for Blair is 0.6876644808984674
The coefficient for Hague is -0.9748271902481128
The coefficient for Europe is -0.22179228698813155
The coefficient for political.knowledge is -0.48344559454795843
The coefficient for gender is 0.015967511560797887
```

Figure 31: LDA - Coefficient values for each column

There are 8 attributes in this dataset and hence there are 8 coefficients. For every one-unit change in x (the different independent variables), y (the dependent variable) changes m (the coefficient value) times. The coefficient values for each column are shown in Figure 31.

The training accuracy of the model is 0.822808671065033.
The test accuracy of the model is 0.8530701754385965.

1.5 Apply KNN Model and Naïve Bayes Model. Interpret the results. (4 marks)

K-Nearest Neighbors

The K-Nearest Neighbors known as KNN is a distance-based machine learning algorithm. It does not construct a model. It is known as a non-parametric method. The classification is computed from a simple majority vote of the nearest neighbors of each point. It is suited for classification where the relationship between features and target classes is numerous, complex and difficult to understand and yet items in a class tend to be fairly homogenous on the values of attributed. It is not suitable if the data is too noisy and the target classes do not have a clear demarcation in terms of attribute values. It can also be used for regression.

The models are built with default parameters and later the best value of k is found to see which suits the best for our data.

The model was built with the default parameters automatically using KNeighborsClassifier() function.

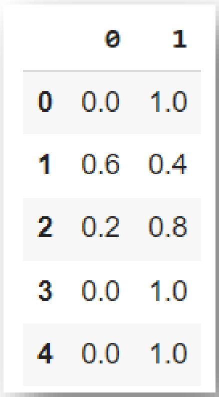
Scaled data

```
KNeighborsClassifier()
```

Figure 32: KNN model built with default parameters

- The **'n_neighbors'** is the number of neighbors to use by default is **5**
- Weights function is used in prediction. The default weight is **'uniform'** where all points in each neighborhood are weighted equally.
- The **'algorithm'** is used to compute the nearest neighbors. The default 'algorithm' value was found to be **auto**.

Table 27: KNN - Probabilities on the test set



	0	1
0	0.0	1.0
1	0.6	0.4
2	0.2	0.8
3	0.0	1.0
4	0.0	1.0

The training accuracy of the model is 0.8680490103675778.
The test accuracy of the model is 0.8442982456140351.

Naïve Bayes

Naïve bayes uses probabilistic models based on Bayes' theorem. It is also a machine learning algorithm. It is called naïve due to the assumption that the features in the dataset are mutually independent. In real world, the independence assumption is often violated, nut naïve Bayes classifiers still tend to perform very well. The idea is to factor all available evidence in form of predictors into the naïve Bayes rule to obtain more accurate probability for class prediction.

Bayes theorem:

$$Posterior = \frac{prior\ x\ likelihood}{evidence}$$

When some of our independent variables are continuous, we cannot calculate conditional probabilities. In Gaussian Naïve Bayes, continuous values associated with each feature (or independent variable) are assumed to be distributed according to a Gaussian distribution. Gaussian Naïve Bayes is also known as Bernoulli Naïve Bayes. This method requires all the features to be of categorical type.

There are no specific parameters in this model like others. Therefore, the model is simply fit with the default parameters using GaussianNB() function.

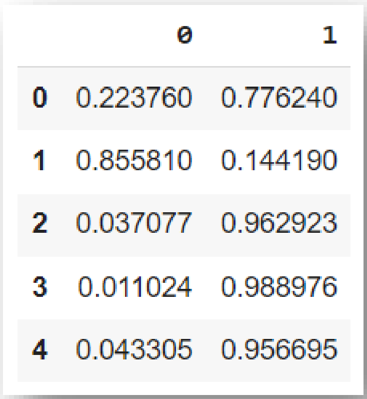
Without scaling the data

```
GaussianNB()
```

Figure 33: Gaussian NB model built with default parameters

- prior probabilities of the classes. If specified the priors are not adjusted according to the data.
- **'var_smoothing'** is the portion of the largest variance of all features that is added to variances for calculation stability. The default 'var_smoothing' is **1e-9**.

Table 28: Gaussian NB - Probabilities on the test set



	0	1
0	0.223760	0.776240
1	0.855810	0.144190
2	0.037077	0.962923
3	0.011024	0.988976
4	0.043305	0.956695

The training accuracy of the model is 0.8199811498586239.
The test accuracy of the model is 0.8574561403508771.

1.6 Model Tuning, Bagging (Random Forest should be applied for Bagging), and Boosting. (7 marks)

Model Tuning

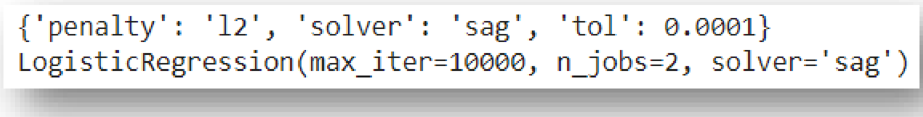
Tuning is the process of increasing the performance of the model without overfitting the data. This is accomplished by passing different 'hyper-parameters' and selecting the best parameters for the model.

Grid search cross validation or GridSearchCV is one of the most common methods of optimizing the parameters.

Many combinations of parameters are passed using the Grid Search CV function. The performance of each combination of these parameters is then evaluated.

Logistic Regression using GridSearchCV

Logistic Regression model was performed using the GridSeachCV. Many combinations of parameters were tried using Grid Search Cross Validation or the Grid Search CV function. Multiple values were passed for the different parameters. Figure 34 shows the best parameters for logistic regression model.



```
{'penalty': 'l2', 'solver': 'sag', 'tol': 0.0001}
LogisticRegression(max_iter=10000, n_jobs=2, solver='sag')
```

Figure 34: Best parameters for Logistic Regression Model

- The **'solver'** is the algorithm used in the process of backpropagation to calculate the weights of the coefficients in the logistic regression model. The optimum 'solver' was found to be **'sag'** solver.
- Penalized logistic regression imposes a penalty to the logistic model for having too many variables. The **'penalty'** is **'l2'.**
- The **'tol'** parameter is the threshold level. The lower the threshold, higher the accuracy and lesser the number of times the model will execute and vice versa. The optimum threshold value was found to be **'0.00001'**.
- The **'max_iter'** parameter is the maximum number of iterations to update the synaptic weights of neurons. The 'max_iter' value was found to be **10000**.

There are other parameters as well but we will use these for grid search and default values for the rest which will be automatically calculated when the model is built.

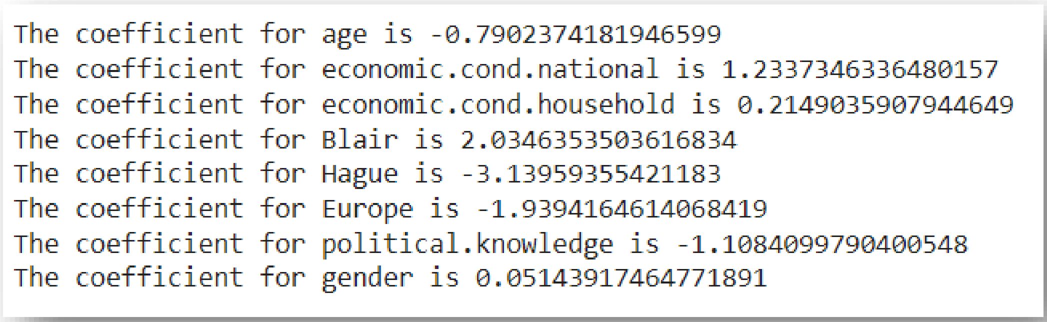

```
The coefficient for age is -0.7902374181946599
The coefficient for economic.cond.national is 1.2337346336480157
The coefficient for economic.cond.household is 0.2149035907944649
The coefficient for Blair is 2.0346353503616834
The coefficient for Hague is -3.13959355421183
The coefficient for Europe is -1.9394164614068419
The coefficient for political.knowledge is -1.1084099790400548
The coefficient for gender is 0.05143917464771891
```

Figure 35: Coefficient values for each column - Logistic Regression using GridSearchCV

There are 6 attributes in this dataset and hence there are 6 coefficients. For every one-unit change in x (the different independent variables), y (the dependent variable) changes m (the coefficient value) times. The coefficient values for each column are shown in Figure 35.

Linear Discriminant Analysis using GridSearchCV

Linear Discriminant Analysis was performed using the GridSeachCV. Many combinations of parameters were tried using Grid Search Cross Validation or the Grid Search CV function. Multiple values were passed for the different parameters. Figure 36 shows the best parameters for Linear Discriminant Analysis model.

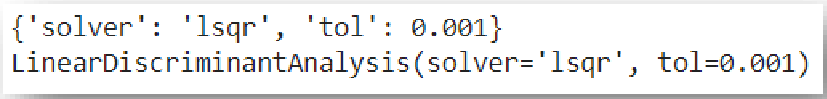
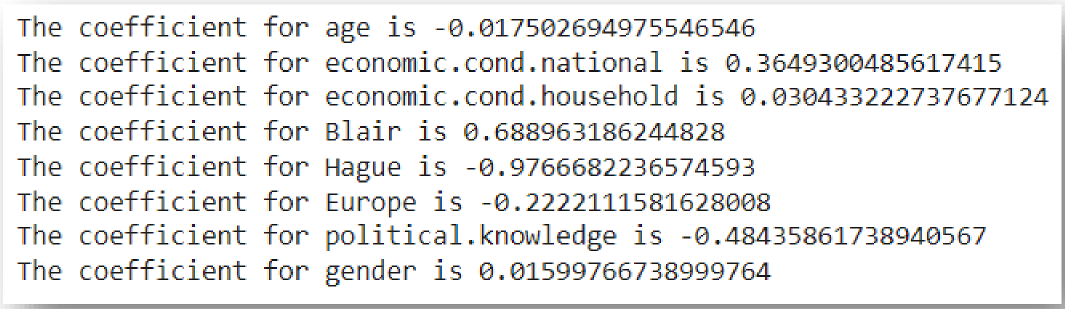

```
{'solver': 'lsqr', 'tol': 0.001}
LinearDiscriminantAnalysis(solver='lsqr', tol=0.001)
```

Figure 36: Best parameters for Linear Discriminant Analysis

- The **'solver'** is the algorithm used in the process of backpropagation to calculate the weights of the coefficients in the logistic regression model. The optimum 'solver' was found to be **'lsqr'** solver.
- The **'tol'** parameter is the threshold level. The lower the threshold, higher the accuracy and lesser the number of times the model will execute and vice versa. The optimum threshold value was found to be **'0.001'**.

There are other parameters as well but we will use these for grid search and default values for the rest which will be automatically calculated when the model is built.



```
The coefficient for age is -0.017502694975546546
The coefficient for economic.cond.national is 0.3649300485617415
The coefficient for economic.cond.household is 0.030433222737677124
The coefficient for Blair is 0.688963186244828
The coefficient for Hague is -0.9766682236574593
The coefficient for Europe is -0.2222111581628008
The coefficient for political.knowledge is -0.48435861738940567
The coefficient for gender is 0.015997667389997640
```

Figure 37: Coefficient values for each column - Linear Discriminant Analysis using GridSearchCV

There are 6 attributes in this dataset and hence there are 6 coefficients. For every one-unit change in x (the different independent variables), y (the dependent variable) changes m (the coefficient value) times. The coefficient values for each column are shown in Figure 37.

KNN model with the best value of K

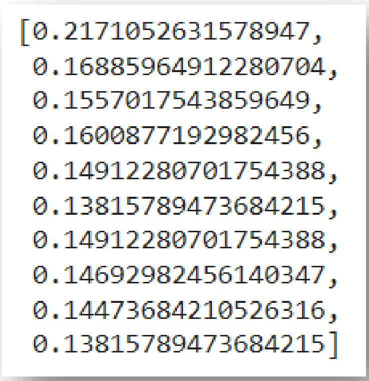
N-neighbors is the k value. The k value should not be too low or too high. Small k values result is very few neighbors and high k values will result the data point to fall under one category and therefore extremely smoothened result with no predictive power. The k value cannot be an even number as it will result in the risk of tie in the decision. The default k value used when building models is 5.

KNN model was performed with different values (odd numbers) for the n_neighbors. The optimum number of neighbors is then found using the mis classification error.

Mis Classification Error (MCE) = 1- Test Accuracy score

The model with the lowest mis classification error score is the best value for k.

Here, the k values (odd numbers) in the range of 1-20 were passed and the mis classification error was calculated as shown in Figure 38 and also plotted as shown in Figure 39.



```
[0.2171052631578947,
 0.16885964912280704,
 0.1557017543859649,
 0.1600877192982456,
 0.14912280701754388,
 0.13815789473684215,
 0.14912280701754388,
 0.14692982456140347,
 0.14473684210526316,
 0.13815789473684215]
```

Figure 38: Mis Classification Error scores

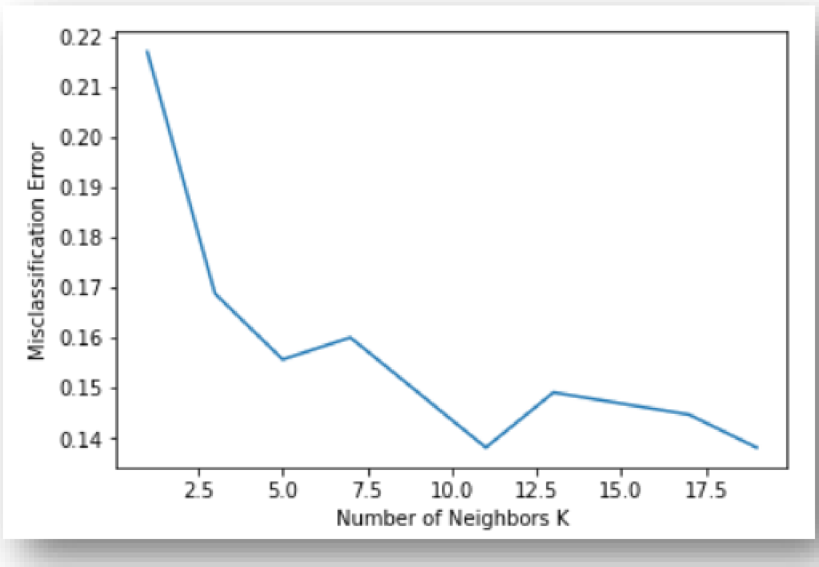
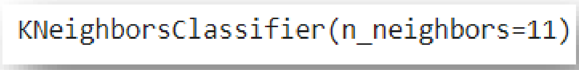


Figure 39: Graph showing the mis classification error for different k values

From Figure 38 and Figure 39, it is seen that the mis classification error value is the lowest when the k value is 11 and 19. We will consider the lower value i.e., 11 as the number of neighbors (k value).

Mis classification error value for k value 11 = 0.13815789473684215

The k value that is found now is then used to build the KNN model. There are other parameters as well but we will define only the k value and default values for the rest which will be automatically calculated when the model is built.



```
KNeighborsClassifier(n_neighbors=11)
```

Figure 40: Best Parameters for KNN model

Boosting:

Boosting is an ensemble learning method that combines a set of weak learners into a single strong learner to minimize training errors. In boosting, a random sample of data is selected, fitted with a model and then **trained sequentially**—that is, each model tries to compensate for the weaknesses of its predecessor. Result from one learner becomes the input for the other and so on, thus improving the model. Boosting uses simple models and tries to 'boost' their aggregate complexity.

There are different types of boosting:

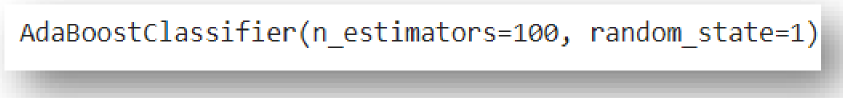
1. Ada Boosting
2. Gradient Boosting

Ada Boosting

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most suited and most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees with only 1 split.

It fits a sequence of weak learners on different weighted training data. It starts by predicting original data set and gives equal weight to each observation. If prediction is incorrect using the first learner, then it gives higher weight to observation which have been predicted incorrectly. Being an iterative process, it continues to add learner(s) until a limit is reached in the number of models or accuracy. We can use AdaBoost algorithms for both classification and regression problem.

The model was built different parameters manually and the best model is chosen.

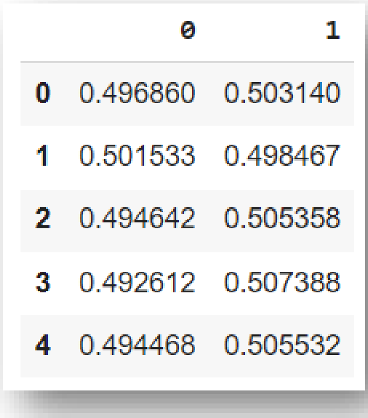


```
AdaBoostClassifier(n_estimators=100, random_state=1)
```

Figure 41: Ada Boosting Model built with best parameters

- The **'n_estimators'** is the number of trees you want to build and **100** was used.
- The default algorithm **SAMME.R** was used. The SAMME.R algorithm typically converges faster than SAMME, achieving a lower test error with fewer boosting iterations.
- The **base estimator** is from which the boosted ensemble is built. The default base estimator is '**None'**. 'None' means that the base estimator is **DecisionTreeClassifier** initialized with **max_depth=1**.

Table 29: Ada Boosting - Probabilities on the Test set



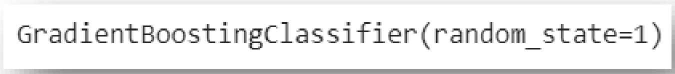
	0	1
0	0.496860	0.503140
1	0.501533	0.498467
2	0.494642	0.505358
3	0.492612	0.507388
4	0.494468	0.505532

Gradient Boosting

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

Therefore, Gradient Boosting Algorithm is generally used when we want to decrease the Bias error. Gradient Boosting Algorithm can be used in regression as well as classification problems.

The model was built different parameters manually and the best model is chosen.

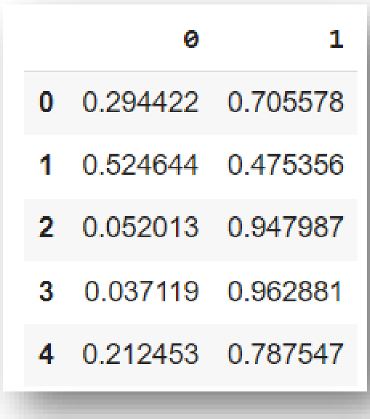


```
GradientBoostingClassifier(random_state=1)
```

Figure 42: Gradient Boosting Model built with best parameters

- The **'n_estimators'** is the number of boosting stages to perform and **100** was used as default.
- The **criterion** is a function to measure the quality of the split. The default value of 'friedman_mse' is generally the best as it can provide a better approximation in some cases.
- The parameter **'min_samples_split'** determines how many observations a node should have for it to be split into left and right child nodes. **2** was found to be the default 'min_samples_split' value.
- The **'min_samples_leaf'** is a parameter that determines how many observations need to be present in each of the terminal nodes or the leaf nodes in all the decision trees in the random forest. In this data, the default 'min_samples_leaf' was determined to be **1**.

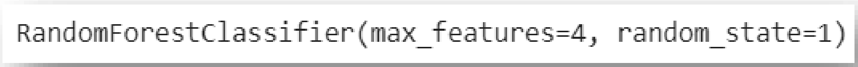
Table 30: Gradient Boosting - Probabilities on the Test set



	0	1
0	0.294422	0.705578
1	0.524644	0.475356
2	0.052013	0.947987
3	0.037119	0.962881
4	0.212453	0.787547

Random Forest

Random Forest technique is an ensemble technique wherein we construct multiple models and take the average output of all the models to take a final decision or make a prediction. Many combinations of parameters were tried manually. Multiple values were passed for the different parameters and the best parameters were chosen.

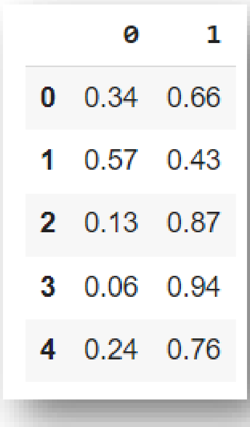


```
RandomForestClassifier(max_features=4, random_state=1)
```

Figure 43: Random Forest Model built with best parameters

- The **'max_depth'** parameter is used to prune the trees. It is the number of decision trees or the level of the trees. The default 'max_depth' parameter is found to be 2 meaning the depth level of the decision tree is **2**.
- The **'max_features'** parameter determines how many number of the independent variables or features a random forest classifier uses for evaluating and splitting the decision nodes. The 'max_features' was found to be **4** in this dataset.
- The **'min_samples_leaf'** is a parameter that determines how many observations need to be present in each of the terminal nodes or the leaf nodes in all the decision trees in the random forest. In this data, the default 'min_samples_leaf' was determined to be **1**.
- The parameter **'min_samples_split'** determines how many observations a node should have for it to be split into left and right child nodes. **2** was found to be the default 'min_samples_split' value.
- The **'n_estimators'** is the number of trees you want to build in a random forest and the default value **100** was used.

Table 31: Random Forest - Probabilities on the Test set



	0	1
0	0.34	0.66
1	0.57	0.43
2	0.13	0.87
3	0.06	0.94
4	0.24	0.76

Bagging Model (Using Random Forest Classifier)

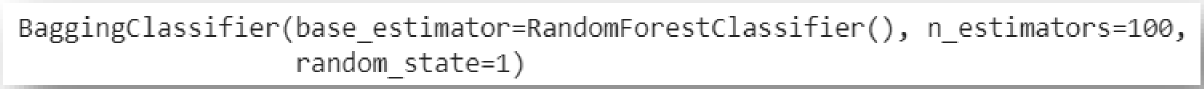
Bagging, also known as bootstrap aggregation, is the ensemble learning method that is commonly used to reduce variance within a noisy dataset. Bagging reduces the chances of overfitting by training each model with a randomly chosen subset of the training data. Training can be done in **parallel**.

In bagging, a random sample of data in a training set is selected with replacement—meaning that the individual data points can be chosen more than once. It is commonly used with tree-based algorithms. Training set for each of the base classifier is independent of each other.

It essentially trains a large number of 'strong' learners in parallel (each model is an over fit for that subset of data). It combines (by averaging or voting) these learners together to 'smooth out' predictions.

Bagging uses complex models and tries to 'smooth out' their predictions.

Here we will use Random Forest as the base classifier to build our bagging model. The model was built different parameters manually and the best model is chosen. There are other parameters as well but we will use default values for the rest which will be automatically calculated when the model is built.



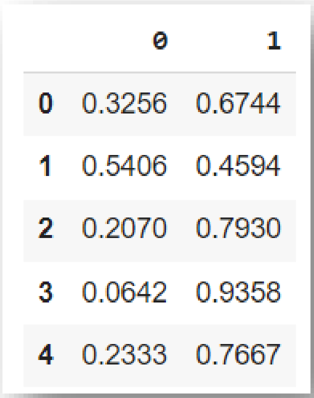
```
BaggingClassifier(base_estimator=RandomForestClassifier(), n_estimators=100,
                  random_state=1)
```

Figure 44: Bagging Model built with Random Forest as base estimator

- The **'n_estimators'** is the number of base estimators in the ensemble and **100** was used.
- The **base estimator** is to fit on random subsets of the dataset. The base estimator **RandomForestClassifier**.
- **Max_features** is the number of features to draw from X to train each base estimator (without replacement by default). The default **max_features=1** is used.

- Max_samples is the number of samples to draw from X to train each base estimator (with replacement by default). The default **max_sample** value of **1** is used.

Table 32: Bagging - Probabilities on the Test set

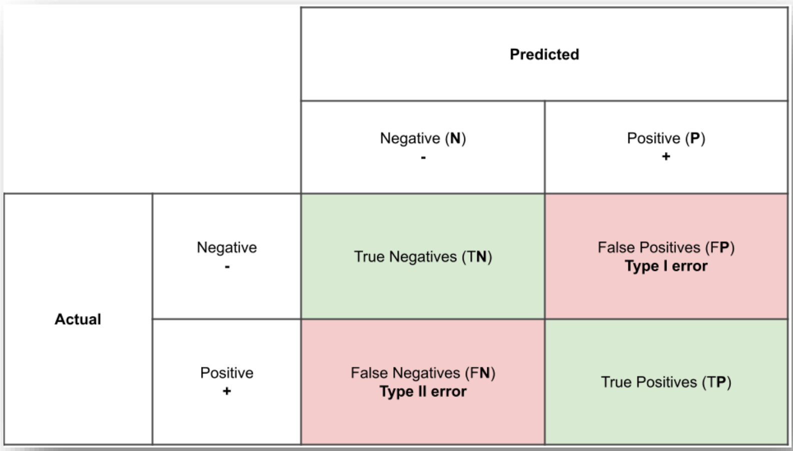


	0	1
0	0.3256	0.6744
1	0.5406	0.4594
2	0.2070	0.7930
3	0.0642	0.9358
4	0.2333	0.7667

1.7 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model. Final Model: Compare the models and write inference which model is best/optimized. (7 marks)

Model Performance Metrics:

Confusion matrix:

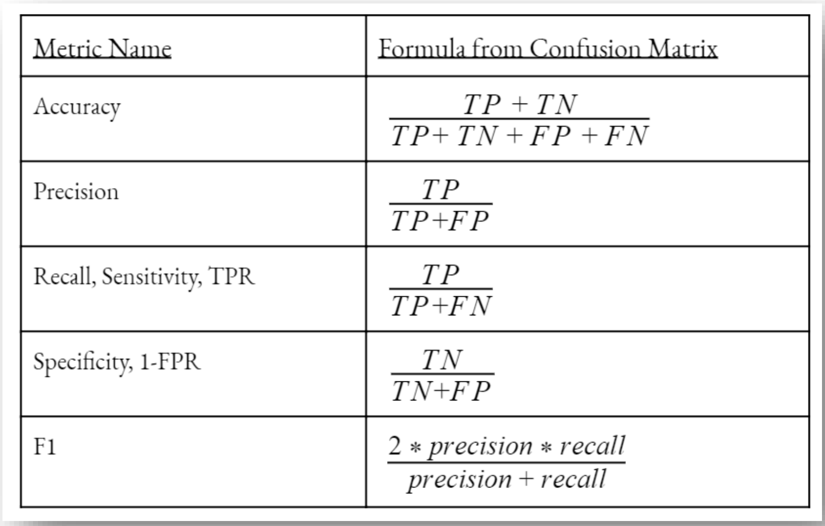


		Predicted	
		Negative (**N**) -	Positive (**P**) +
Actual	Negative -	True Negatives (**TN**)	False Positives (**FP**) **Type I error**
	Positive +	False Negatives (**FN**) **Type II error**	True Positives (**TP**)

Figure 45: Confusion Matrix

A confusion matrix is a 2x2 tabular structure reflecting the performance of the model in four blocks. Figure 45 shows how a confusion matrix will look like. True Positive (TP) and True Negative (TN) are correct predictions. False Positive (FP) and False Negative (FN) are incorrect predictions.

Table 33: Confusion Matrix formulas



Metric Name	Formula from Confusion Matrix
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
Precision	$\frac{TP}{TP+FP}$
Recall, Sensitivity, TPR	$\frac{TP}{TP+FN}$
Specificity, 1-FPR	$\frac{TN}{TN+FP}$
F1	$\frac{2 * precision * recall}{precision + recall}$

- **Accuracy** – it is a measure of how accurately or cleanly the model classifies the data points. Lesser the false predictions, more the accuracy. In a classification problem, the best score is **100%** accuracy.
- **Precision** – it is a measure of how many among the points identified as positive by the model, are really positive. A precision score of 1.0 means that all the points are identified as positive by the model. **1.0** is a perfect precision score. However, it does not indicate about the number of observations that were not labelled correctly.
- **Recall or sensitivity** – is the ratio of correctly predicted positive observations to the all observations in actual class. A perfect recall score is **1.0** but a recall score above 0.5 is considered as a good recall score. However, the recall score does not indicate about how many observations are incorrectly predicted.
- **Specificity** – is a measure of how many of the actual negative data points are identified as negative by the model.
- **F1** – it is the measure of the model's accuracy on a dataset. The F-score is a way of combining the precision and recall of the model and it is defined as the harmonic mean of the model's precision and recall. An F1 score is considered perfect when it's **1**, while the model is a total failure when the score is 0.

All these can be calculated from the confusion matrix by using the formulas given in Table 33.

Classification report:

A Classification report is used to measure the quality of predictions from a classification algorithm. The classification report shows the main classification metrics and their scores. The metrics are precision, recall, f1-score and accuracy for the actual and predicted data. The metrics are calculated by using true and false positives, true and false negatives from the confusion matrix.

ROC Curve:

Receiver Operating Characteristics (ROC) Curve is a technique for visualizing classifier performance. It is a graph between true positive (TP) rate and false positive (FP) rate.

$$\text{TP rate} = \frac{TP}{total\ positive}$$

$$\text{FP rate} = \frac{FP}{total\ negative}$$

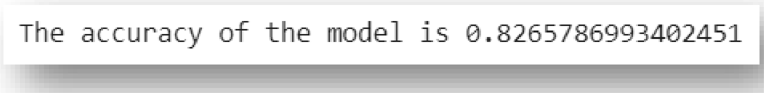
ROC graph is a trade-off between benefits (TP) and costs (FP). The steeper the ROC Curve, the stronger the model will be and vice versa.

ROC_AUC score:

Area under the ROC Curve (AUC) is the measure of the area under the ROC Curve. The ROC_AUC score gives us the value of the area under the ROC Curve. The larger the area under the curve, the better the model.

Logistic Regression

Training Data



```
The accuracy of the model is 0.8265786993402451
```

Figure 46: Accuracy of the Logistic Regression Model for Training Data

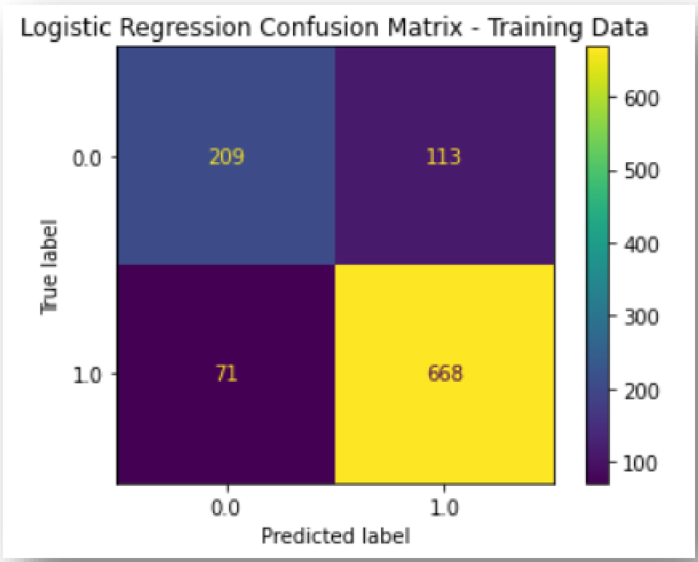


Figure 47: Confusion Matrix of the Logistic Regression Model for Training Data

The confusion matrix was calculated and shown in Figure 47. The accuracy score for the training data was found to be 0.83 which is shown in Figure 46.

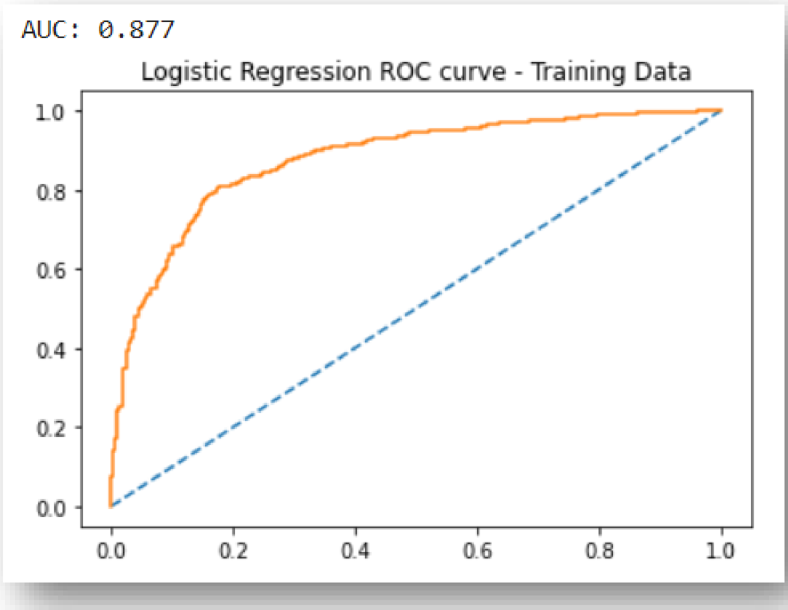
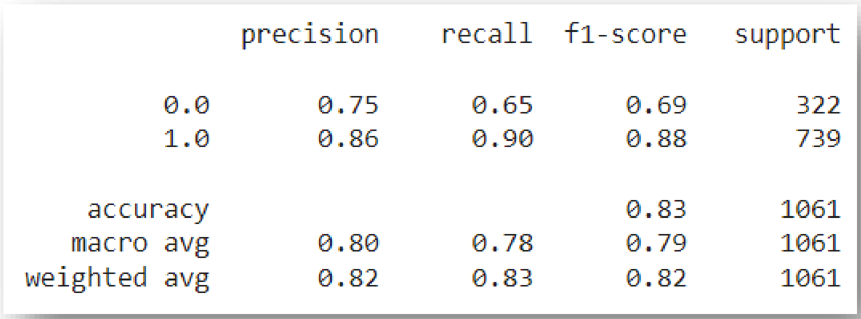


Figure 48: Logistic Regression ROC Curve for Training Data

The ROC_AUC score for the training data was calculated to be 0.877. The ROC curve was plotted and is shown in Figure 48.

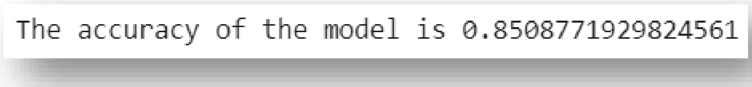


```
              precision    recall  f1-score   support

         0.0       0.75      0.65      0.69       322
         1.0       0.86      0.90      0.88       739

    accuracy                           0.83      1061
   macro avg       0.80      0.78      0.79      1061
weighted avg       0.82      0.83      0.82      1061
```

Figure 49: Logistic Regression Classification Report for Training Data

The classification report for the logistic regression model with the scores for the different model performance measures is calculated and shown in Figure 49. From this figure we can see that the precision of the model is 0.86 means 86% of the data points identified as positive by the model, are really positive. The f1-score is 0.88 means the model is 88% accurate on this data set. The model has 83% accuracy. The recall score is 0.90 which means 90% of the positive observations are correctly predicted.

Test Data



```
The accuracy of the model is 0.8508771929824561
```

Figure 50: Accuracy of the Logistic Regression Model for Test Data

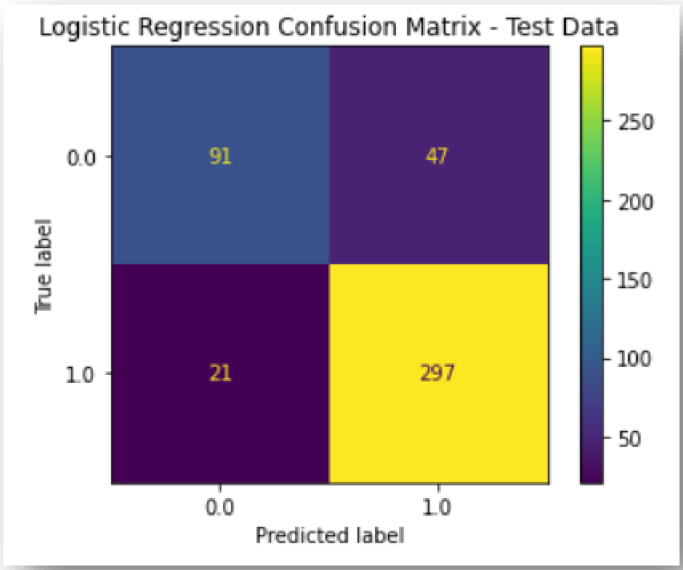


Figure 51: Confusion Matrix of the Logistic Regression Model for Test Data

The confusion matrix was calculated and shown in Figure 51. The accuracy score for the test data was found to be 0.85 which is shown in Figure 50.

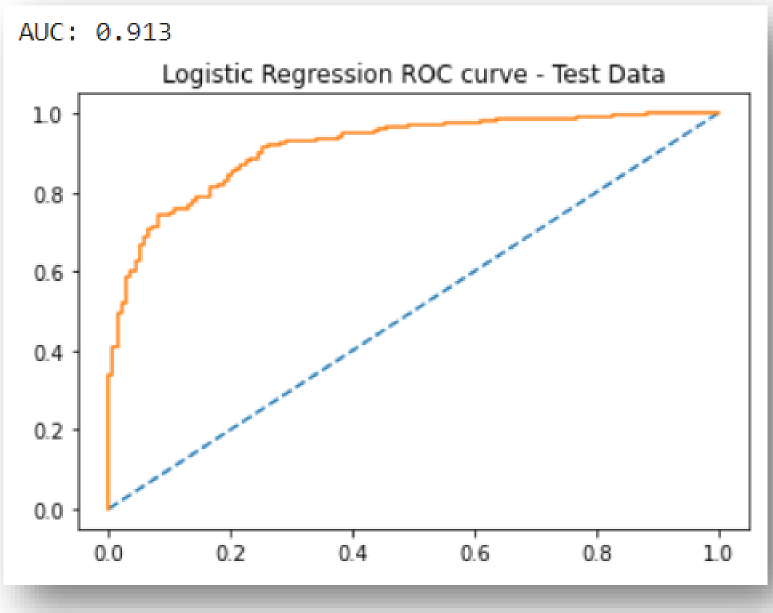
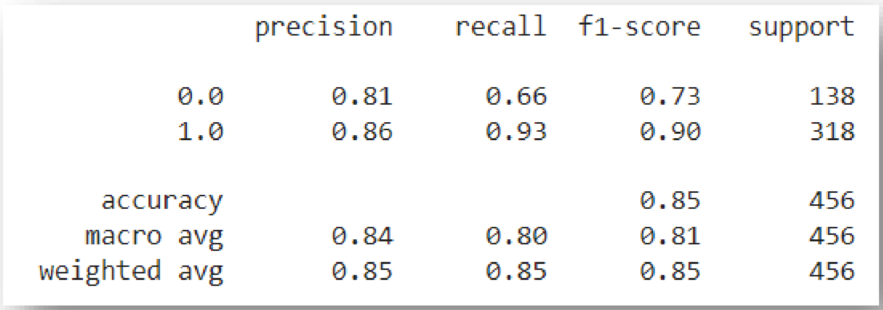


Figure 52: Logistic Regression ROC Curve for Test Data

The ROC_AUC score for the test data was calculated to be 0.913. The ROC curve was plotted and is shown in Figure 52.



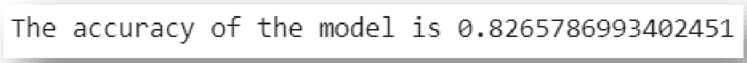
```
              precision    recall  f1-score   support

         0.0       0.81      0.66      0.73       138
         1.0       0.86      0.93      0.90       318

    accuracy                           0.85       456
   macro avg       0.84      0.80      0.81       456
weighted avg       0.85      0.85      0.85       456
```

Figure 53: Logistic Regression Classification Report for Test Data

The classification report for the logistic regression model with the scores for the different model performance measures is calculated and shown in Figure 53. From this figure we can see that the precision of the model is 0.86 means 86% of the data points identified as positive by the model, are really positive. The f1-score is 0.90 means the model is 90% accurate on this data set. The model has 85% accuracy. The recall score is 0.93 which means 93% of the positive observations are correctly predicted.

Logistic Regression using GridSearchCV

Training Data



```
The accuracy of the model is 0.8265786993402451
```

Figure 54: Accuracy of the Logistic Regression Model using Grid Search for Training Data

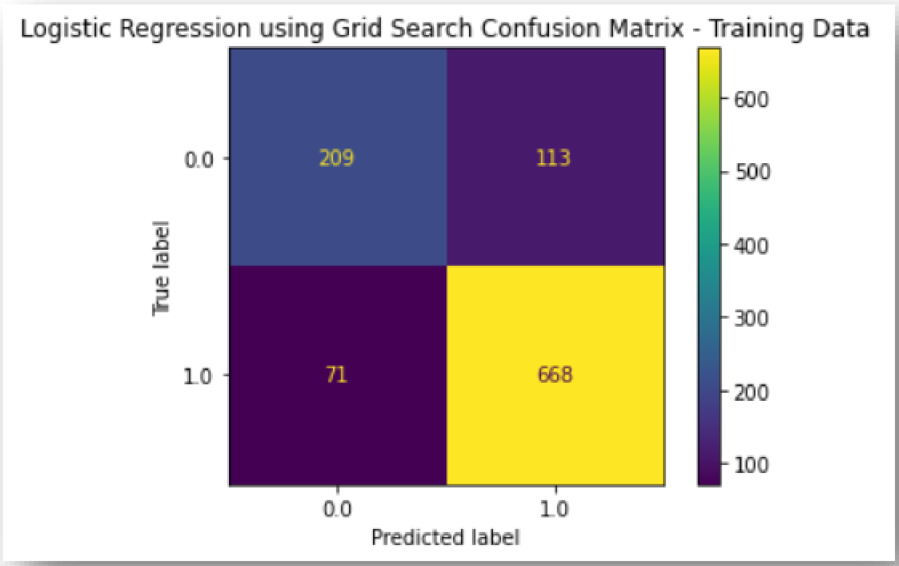


Figure 55: Confusion Matrix of the Logistic Regression Model using Grid Search for Training Data

The confusion matrix was calculated and shown in Figure 55. The accuracy score for the training data was found to be 0.83 which is shown in Figure 54.

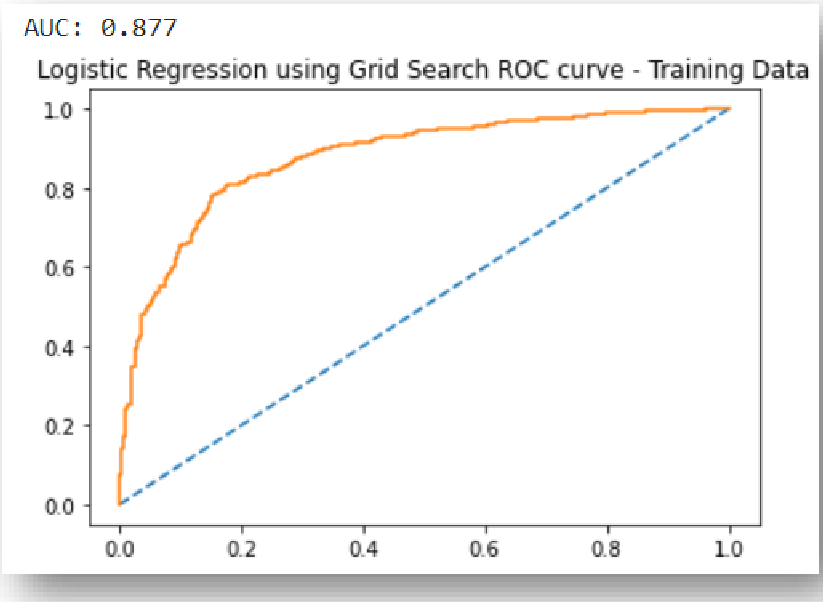

AUC: 0.877

Figure 56: Logistic Regression using Grid Search ROC Curve for Training Data

The ROC_AUC score for the training data was calculated to be 0.877. The ROC curve was plotted and is shown in Figure 56.

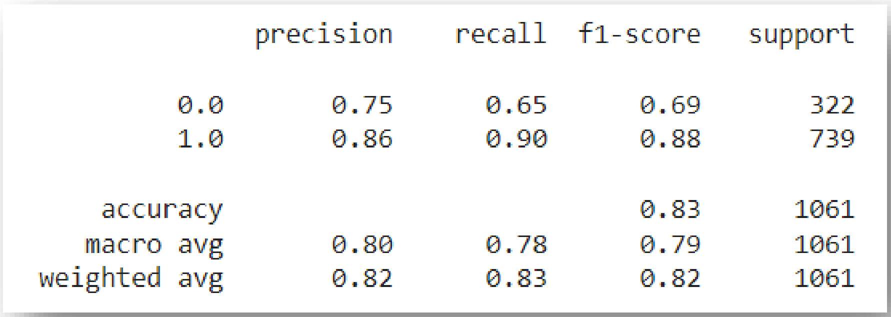
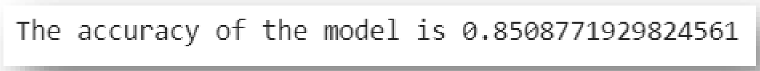

```
              precision    recall  f1-score   support

         0.0       0.75      0.65      0.69       322
         1.0       0.86      0.90      0.88       739

    accuracy                           0.83      1061
   macro avg       0.80      0.78      0.79      1061
weighted avg       0.82      0.83      0.82      1061
```

Figure 57: Logistic Regression using Grid Search Classification Report for Training Data

The classification report for the logistic regression model using Grid Search with the scores for the different model performance measures is calculated and shown in Figure 57. From this figure we can see that the precision of the model is 0.86 means 86% of the data points identified as positive by the model, are really positive. The f1-score is 0.88 means the model is 88% accurate on this data set. The model has 83% accuracy. The recall score is 0.90 which means 90% of the positive observations are correctly predicted.

Test Data



```
The accuracy of the model is 0.8508771929824561
```

Figure 58: Accuracy of the Logistic Regression Model using Grid Search for Test Data

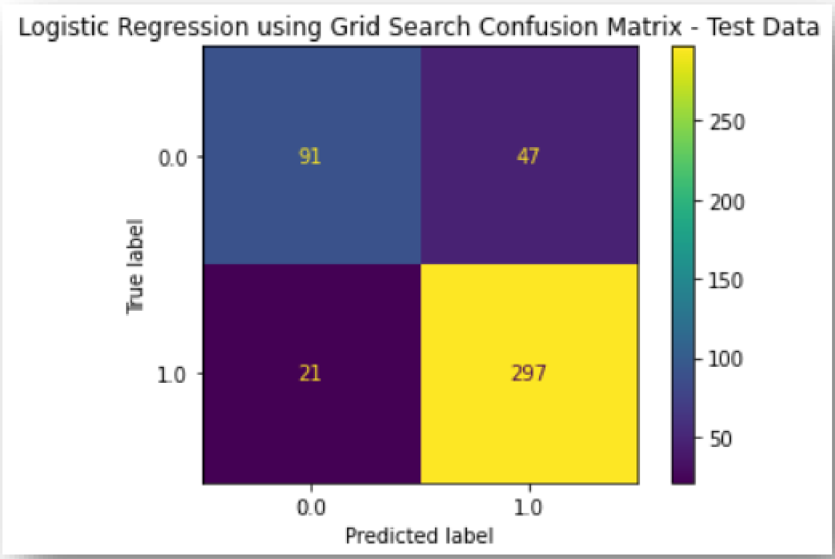


Figure 59: Confusion Matrix of the Logistic Regression Model using Grid Search for Test Data

The confusion matrix was calculated and shown in Figure 59. The accuracy score for the test data was found to be 0.85 which is shown in Figure 58.

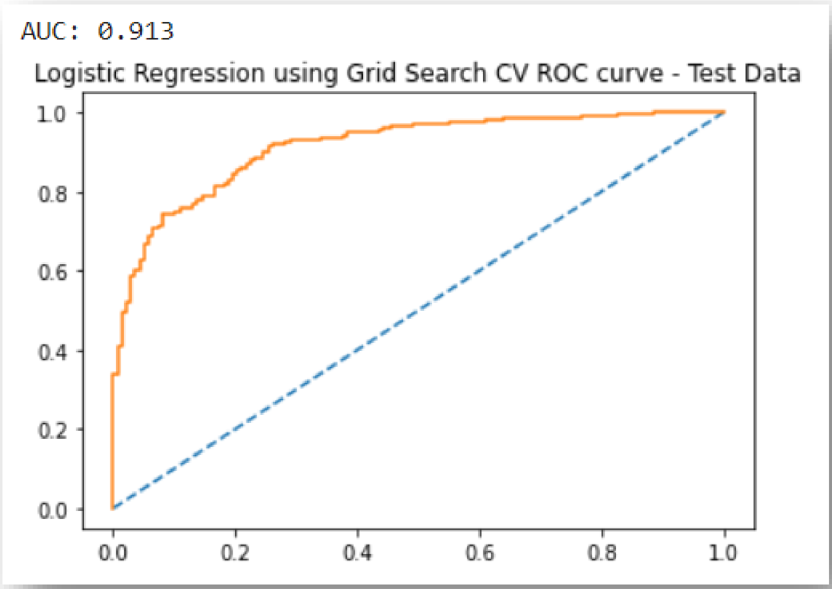
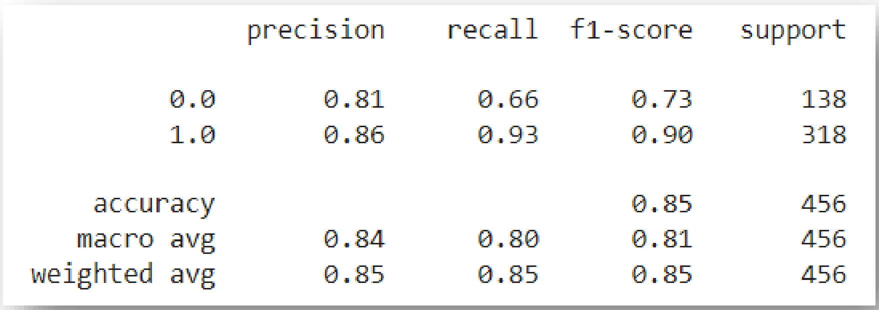


Figure 60: Logistic Regression using Grid Search ROC Curve for Test Data

The ROC_AUC score for the test data was calculated to be 0.913. The ROC curve was plotted and is shown in Figure 60.



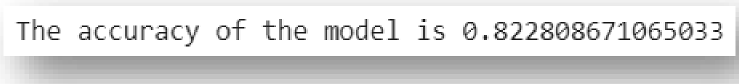
```
              precision    recall  f1-score   support

         0.0       0.81      0.66      0.73       138
         1.0       0.86      0.93      0.90       318

    accuracy                           0.85       456
   macro avg       0.84      0.80      0.81       456
weighted avg       0.85      0.85      0.85       456
```

Figure 61: Logistic Regression using Grid Search Classification Report for Test Data

The classification report for the logistic regression model using Grid Search with the scores for the different model performance measures is calculated and shown in Figure 61. From this figure we can see that the precision of the model is 0.86 means 86% of the data points identified as positive by the model, are really positive. The f1-score is 0.90 means the model is 90% accurate on this data set. The model has 85% accuracy. The recall score is 0.93 which means 93% of the positive observations are correctly predicted.

Linear Discriminant Analysis

Training Data



```
The accuracy of the model is 0.822808671065033
```

Figure 62: Accuracy of the LDA Model for Training Data

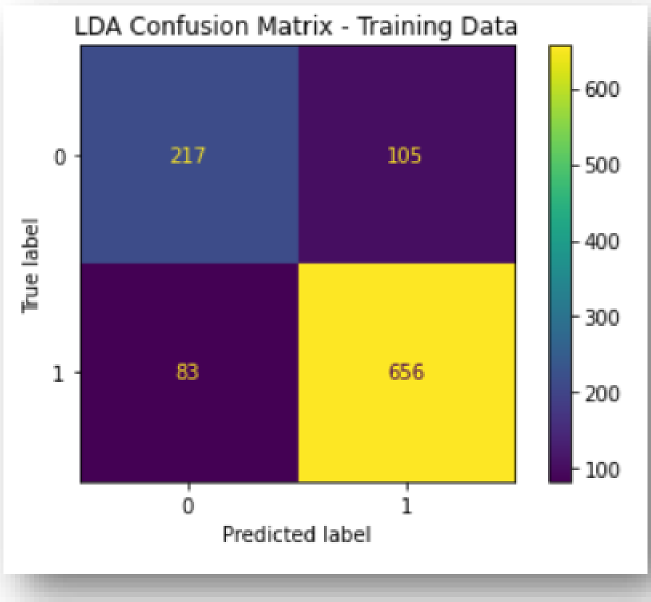


Figure 63: Confusion Matrix of the LDA Model for Training Data

The confusion matrix was calculated and shown in Figure 63. The accuracy score for the training data was found to be 0.82 which is shown in Figure 62.

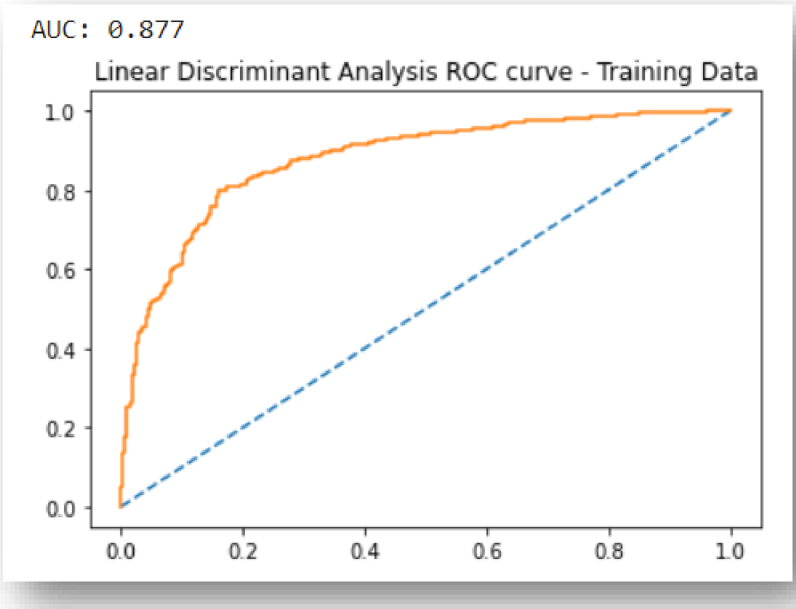
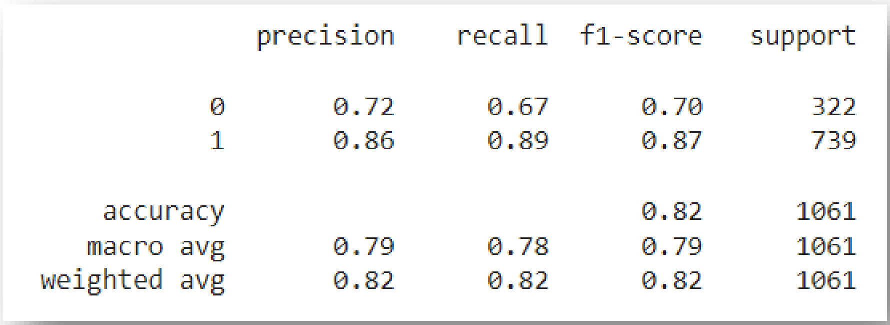


Figure 64: Linear Discriminant Analysis ROC Curve for Training Data

The ROC_AUC score for the train data was calculated to be 0.877. The ROC curve was plotted and is shown in Figure 64.

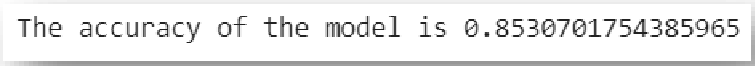


```
              precision    recall  f1-score   support

           0       0.72      0.67      0.70       322
           1       0.86      0.89      0.87       739

    accuracy                           0.82      1061
   macro avg       0.79      0.78      0.79      1061
weighted avg       0.82      0.82      0.82      1061
```

Figure 65: Linear Discriminant Analysis Classification Report for Training Data

The classification report for the Linear Discriminant Analysis with the scores for the different model performance measures is calculated and shown in Figure 65. From this figure we can see that the precision of the model is 0.86 means 86% of the data points identified as positive by the model, are really positive. The f1-score is 0.87 means the model is 87% accurate on this data set. The model has 82% accuracy. The recall score is 0.89 which means 89% of the positive observations are correctly predicted.

Test Data



```
The accuracy of the model is 0.8530701754385965
```

Figure 66: Accuracy of the LDA Model for Test Data

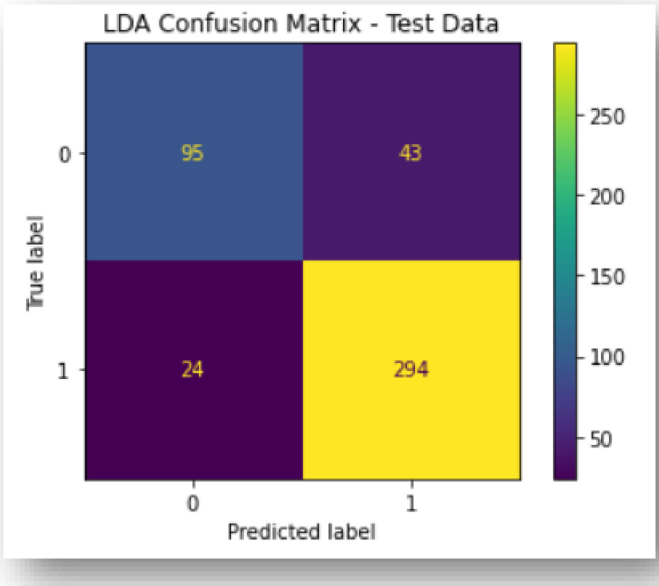


Figure 67: Confusion Matrix of the Linear Discriminant Model for Test Data

The confusion matrix was calculated and shown in Figure 67. The accuracy score for the test data was found to be 0.85 which is shown in Figure 66.

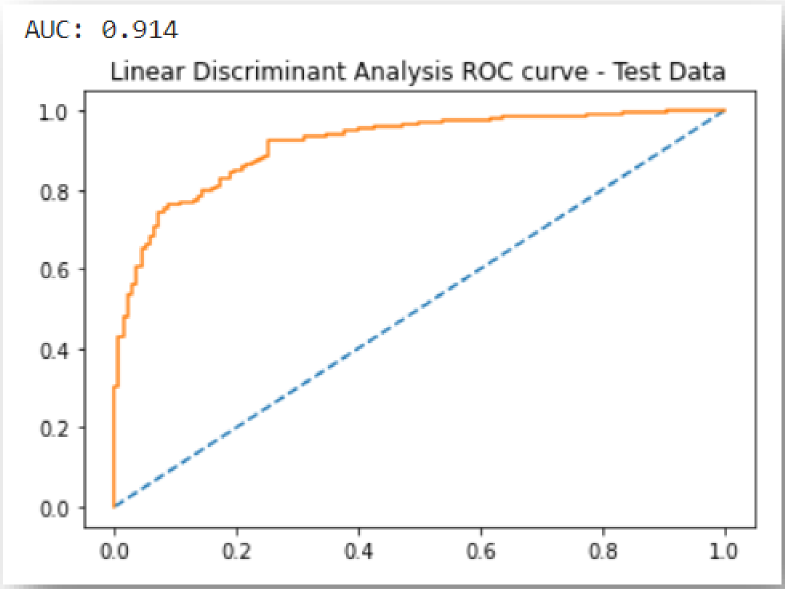
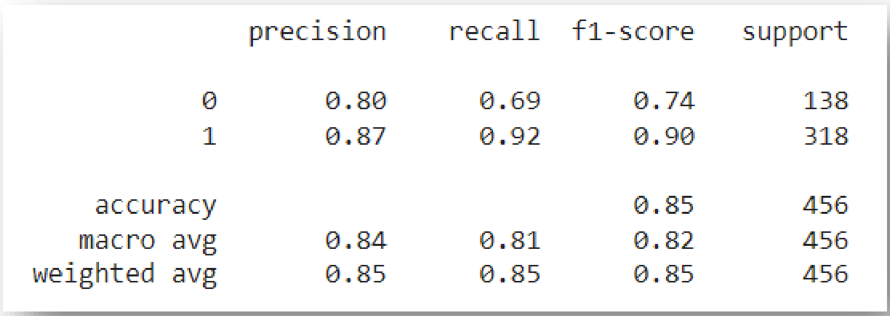


Figure 68: Linear Discriminant Analysis ROC Curve for Test Data

The ROC_AUC score for the test data was calculated to be 0.914. The ROC curve was plotted and is shown in Figure 68.



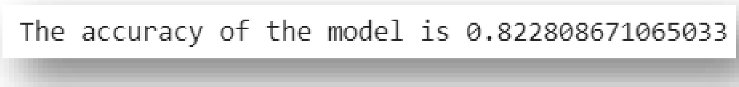
```
              precision    recall  f1-score   support

           0       0.80      0.69      0.74       138
           1       0.87      0.92      0.90       318

    accuracy                           0.85       456
   macro avg       0.84      0.81      0.82       456
weighted avg       0.85      0.85      0.85       456
```

Figure 69: Linear Discriminant Analysis Classification Report for Test Data

The classification report for the Linear Discriminant Analysis with the scores for the different model performance measures is calculated and shown in Figure 69. From this figure we can see that the precision of the model is 0.87 means 87% of the data points identified as positive by the model, are really positive. The f1-score is 0.90 means the model is 90% accurate on this data set. The model has 85% accuracy. The recall score is 0.92 which means 92% of the positive observations are correctly predicted.

Linear Discriminant Analysis using GridSearchCV

Training Data



```
The accuracy of the model is 0.822808671065033
```

Figure 70: Accuracy of the LDA Model using Grid Search for Training Data

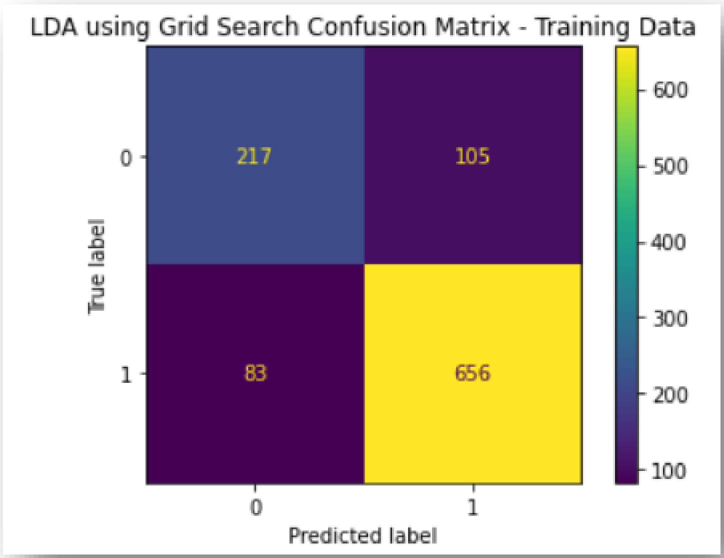


Figure 71: Confusion Matrix of the Linear Discriminant Analysis using Grid Search for Training Data

The confusion matrix was calculated and shown in Figure 71. The accuracy score for the training data was found to be 0.82 which is shown in Figure 70.

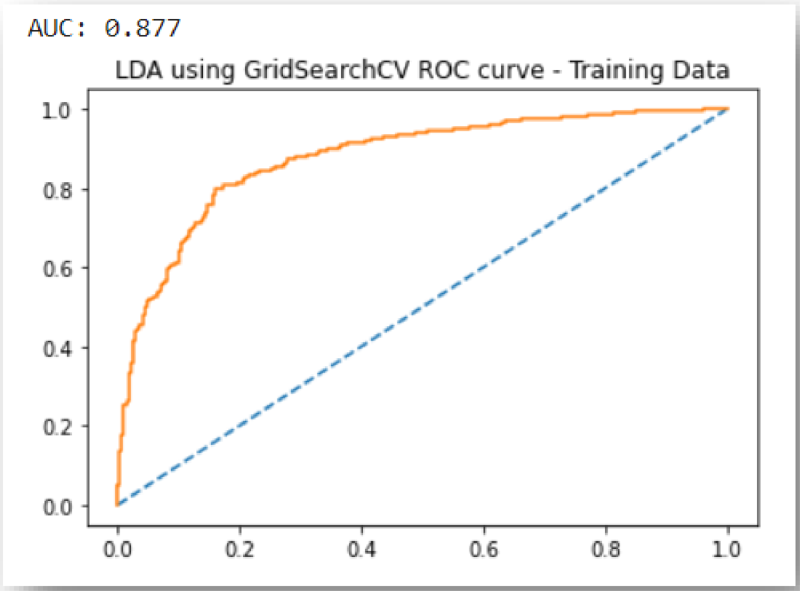

```
AUC: 0.877
```

Figure 72: Linear Discriminant Analysis using Grid Search ROC Curve for Training Data

The ROC_AUC score for the train data was calculated to be 0.877. The ROC curve was plotted and is shown in Figure 72.

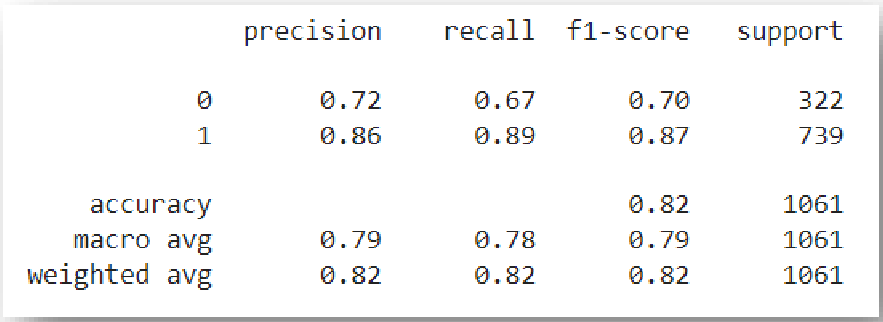
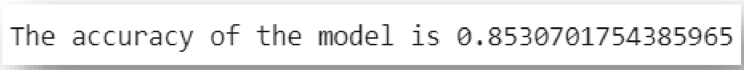

```
              precision    recall  f1-score   support

           0       0.72      0.67      0.70       322
           1       0.86      0.89      0.87       739

    accuracy                           0.82      1061
   macro avg       0.79      0.78      0.79      1061
weighted avg       0.82      0.82      0.82      1061
```

Figure 73: Linear Discriminant Analysis using Grid Search Classification Report for Training Data

The classification report for the Linear Discriminant Analysis using Grid Search with the scores for the different model performance measures is calculated and shown in Figure 73. From this figure we can see that the precision of the model is 0.86 means 86% of the data points identified as positive by the model, are really positive. The f1-score is 0.87 means the model is 87% accurate on this data set. The model has 82% accuracy. The recall score is 0.89 which means 89% of the positive observations are correctly predicted.

Test Data



```
The accuracy of the model is 0.8530701754385965
```

Figure 74: Accuracy of the LDA Model using Grid Search for Test Data

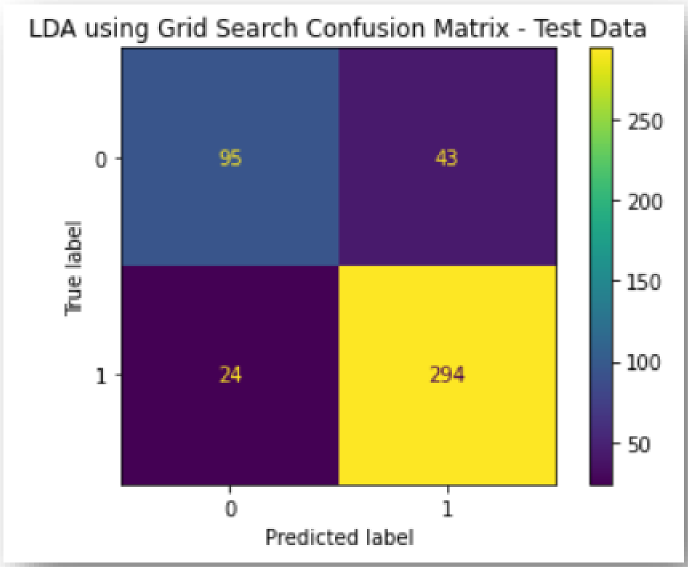


Figure 75: Confusion Matrix of the Linear Discriminant Analysis using Grid Search for Test Data

The confusion matrix was calculated and shown in Figure 75. The accuracy score for the training data was found to be 0.85 which is shown in Figure 74.

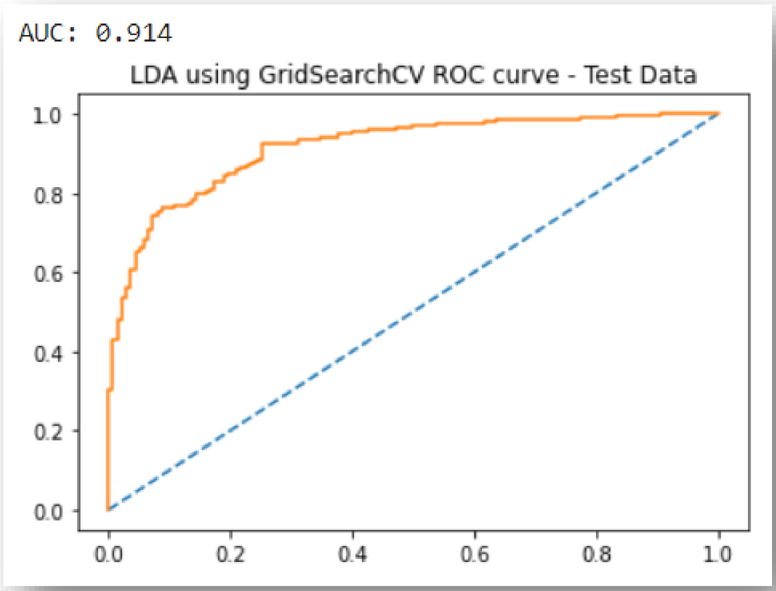
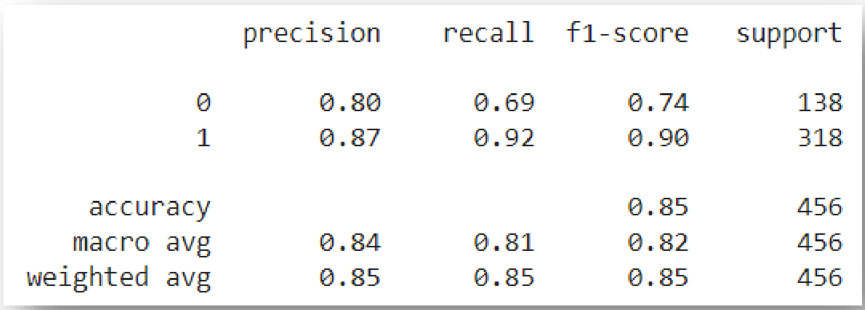


Figure 76: Linear Discriminant Analysis using Grid Search ROC Curve for Test Data

The ROC_AUC score for the test data was calculated to be 0.914. The ROC curve was plotted and is shown in Figure 76.



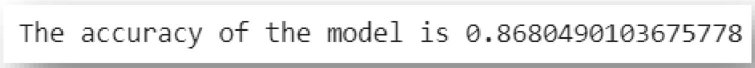
```
              precision    recall  f1-score   support

           0       0.80      0.69      0.74       138
           1       0.87      0.92      0.90       318

    accuracy                           0.85       456
   macro avg       0.84      0.81      0.82       456
weighted avg       0.85      0.85      0.85       456
```

Figure 77: Linear Discriminant Analysis using Grid Search Classification Report for Test Data

The classification report for the Linear Discriminant Analysis using Grid Search with the scores for the different model performance measures is calculated and shown in Figure 77. From this figure we can see that the precision of the model is 0.87 means 87% of the data points identified as positive by the model, are really positive. The f1-score is 0.90 means the model is 90% accurate on this data set. The model has 85% accuracy. The recall score is 0.92 which means 92% of the positive observations are correctly predicted.

KNN Model

Training Data



```
The accuracy of the model is 0.8680490103675778
```

Figure 78: Accuracy of the KNN Model for Training Data

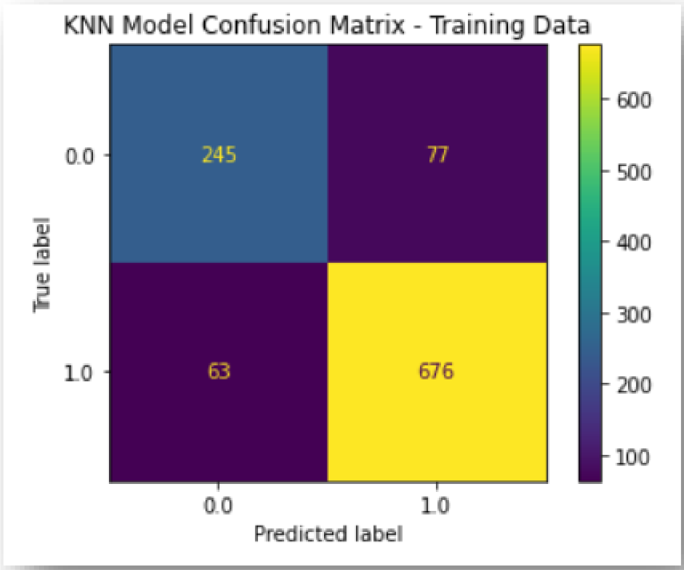


Figure 79: Confusion Matrix of the KNN Model for Training Data

The confusion matrix was calculated and shown in Figure 79. The accuracy score for the training data was found to be 0.87 which is shown in Figure 78.

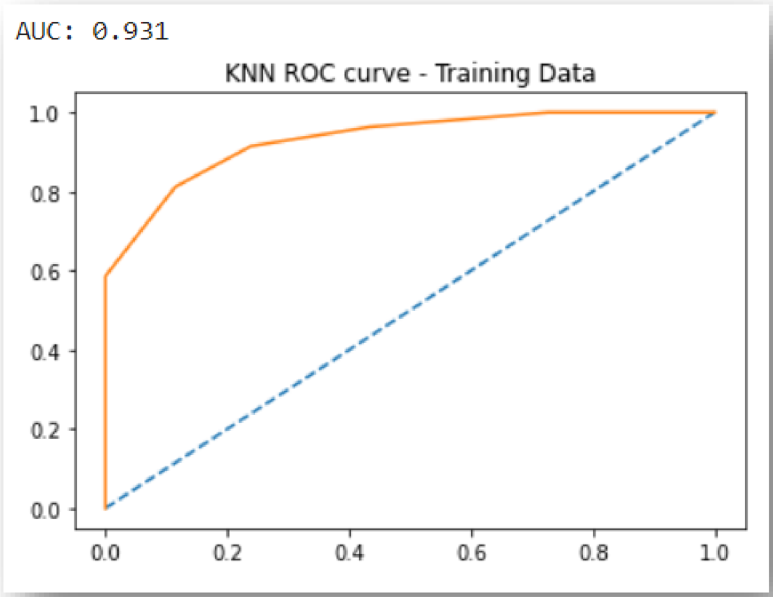
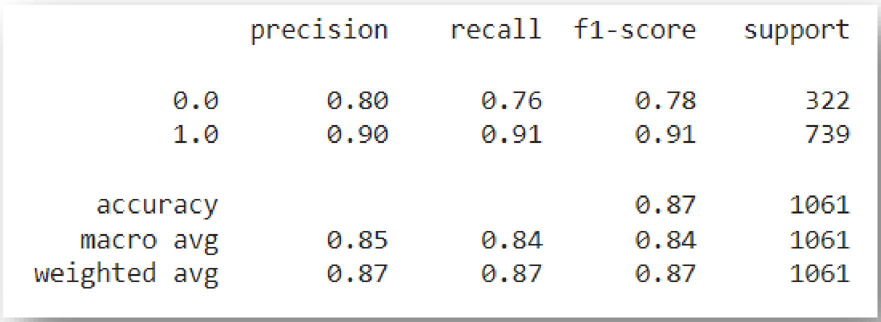


Figure 80: KNN Model ROC Curve for Training Data

The ROC_AUC score for the training data was calculated to be 0.931. The ROC curve was plotted and is shown in Figure 80.

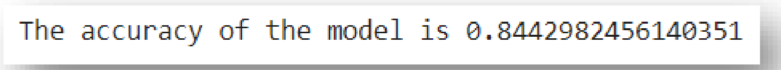


```
              precision    recall  f1-score   support

         0.0       0.80      0.76      0.78       322
         1.0       0.90      0.91      0.91       739

    accuracy                           0.87      1061
   macro avg       0.85      0.84      0.84      1061
weighted avg       0.87      0.87      0.87      1061
```

Figure 81: KNN Model Classification Report for Training Data

The classification report for the KNN Model with the scores for the different model performance measures is calculated and shown in Figure 81. From this figure we can see that the precision of the model is 0.90 means 90% of the data points identified as positive by the model, are really positive. The f1-score is 0.91 means the model is 91% accurate on this data set. The model has 87% accuracy. The recall score is 0.91 which means 91% of the positive observations are correctly predicted.

Test Data



```
The accuracy of the model is 0.8442982456140351
```

Figure 82: Accuracy of the KNN Model for Test Data

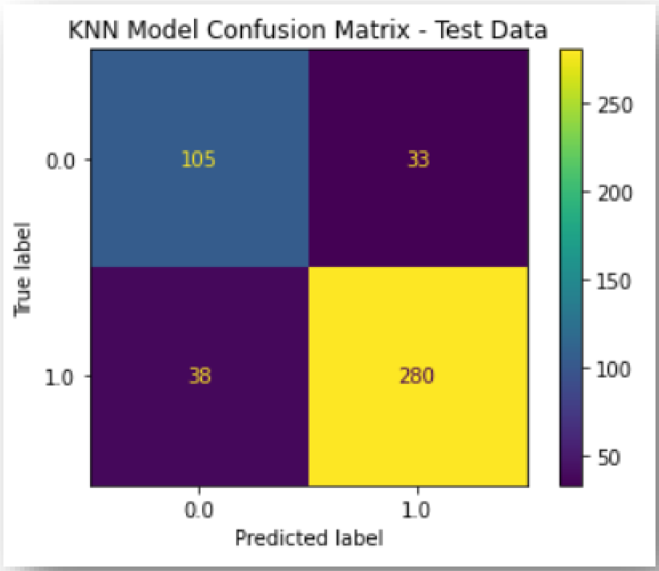


Figure 83: Confusion Matrix of the KNN Model for Test Data

The confusion matrix was calculated and shown in Figure 83. The accuracy score for the test data was found to be 0.84 which is shown in Figure 82.

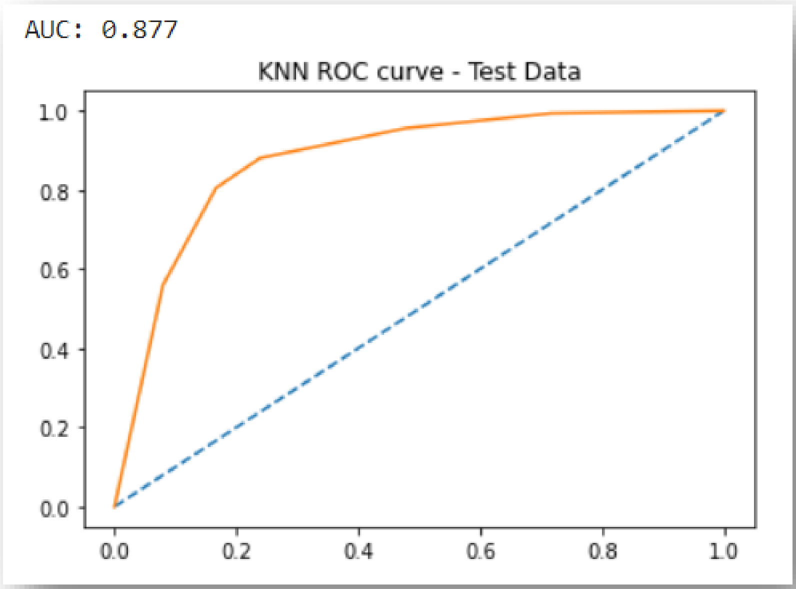
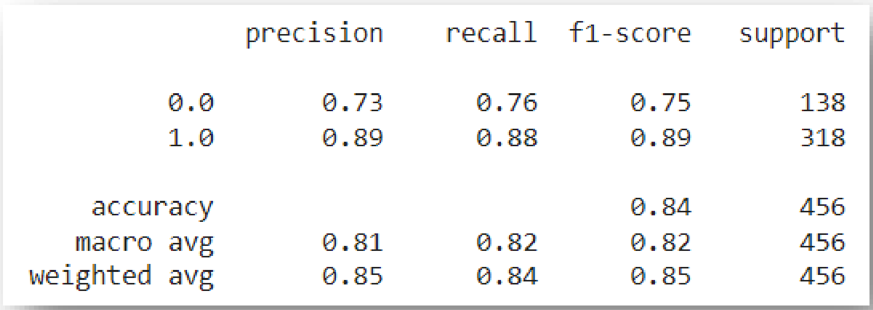


Figure 84: KNN Model ROC Curve for Test Data

The ROC_AUC score for the test data was calculated to be 0.877. The ROC curve was plotted and is shown in Figure 84.



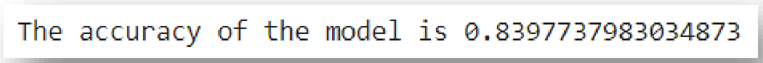
```
              precision    recall  f1-score   support

         0.0       0.73      0.76      0.75       138
         1.0       0.89      0.88      0.89       318

    accuracy                           0.84       456
   macro avg       0.81      0.82      0.82       456
weighted avg       0.85      0.84      0.85       456
```

Figure 85: KNN Model Classification Report for Test Data

The classification report for the KNN Model with the scores for the different model performance measures is calculated and shown in Figure 85. From this figure we can see that the precision of the model is 0.89 means 89% of the data points identified as positive by the model, are really positive. The f1-score is 0.89 means the model is 89% accurate on this data set. The model has 84% accuracy. The recall score is 0.88 which means 88% of the positive observations are correctly predicted.

KNN model with the best value of K

Training Data



```
The accuracy of the model is 0.8397737983034873
```

Figure 86: Accuracy of the KNN Model with best value of k using for Training Data

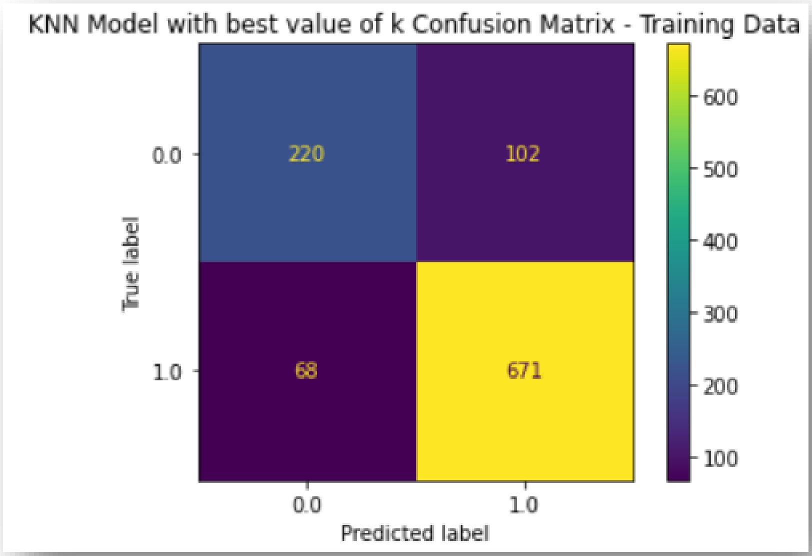


Figure 87: Confusion Matrix of the KNN Model with best value of k for Training Data

The confusion matrix was calculated and shown in Figure 87. The accuracy score for the training data was found to be 0.84 which is shown in Figure 86.

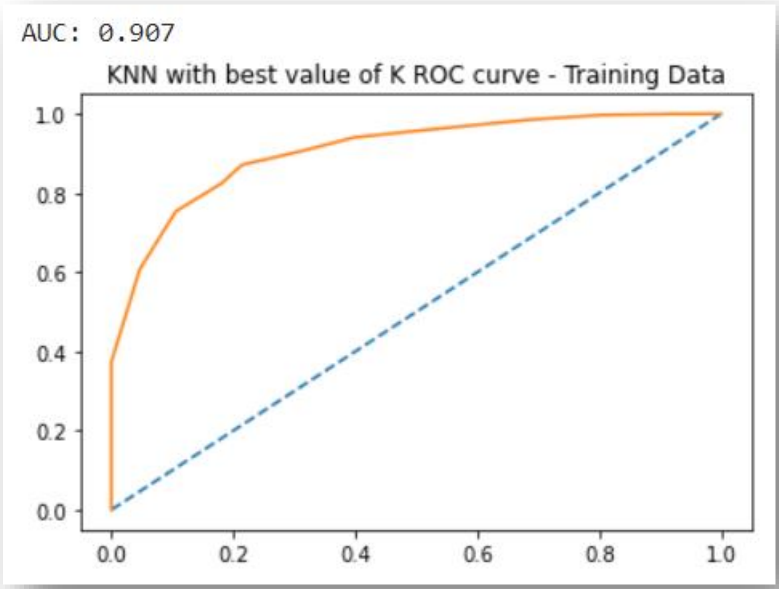
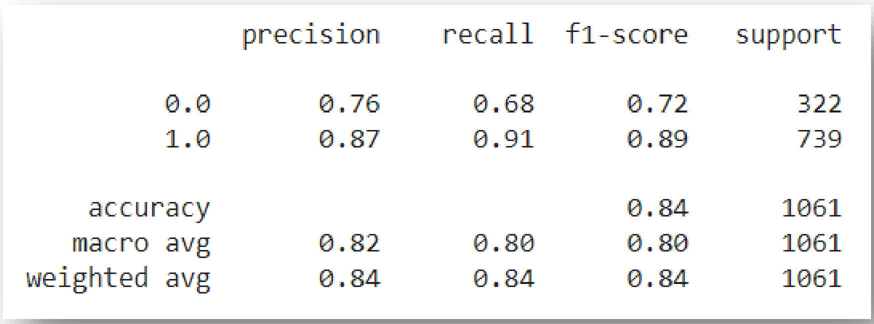


Figure 88: KNN Model with best value of K ROC Curve for Training Data

The ROC_AUC score for the training data was calculated to be 0.907. The ROC curve was plotted and is shown in Figure 88.

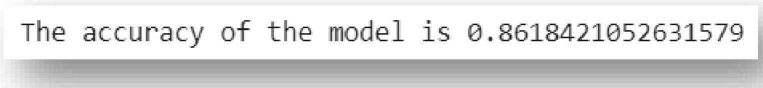


```
              precision    recall  f1-score   support

         0.0       0.76      0.68      0.72       322
         1.0       0.87      0.91      0.89       739

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

Figure 89: KNN Model with the best value of K Classification Report for Training Data

The classification report for the KNN Model with the best value of K with the scores for the different model performance measures is calculated and shown in Figure 89. From this figure we can see that the precision of the model is 0.87 means 87% of the data points identified as positive by the model, are really positive. The f1-score is 0.89 means the model is 89% accurate on this data set. The model has 84% accuracy. The recall score is 0.91 which means 91% of the positive observations are correctly predicted.

Test Data



```
The accuracy of the model is 0.8618421052631579
```

Figure 90: Accuracy of the KNN Model with best value of k using for Test Data

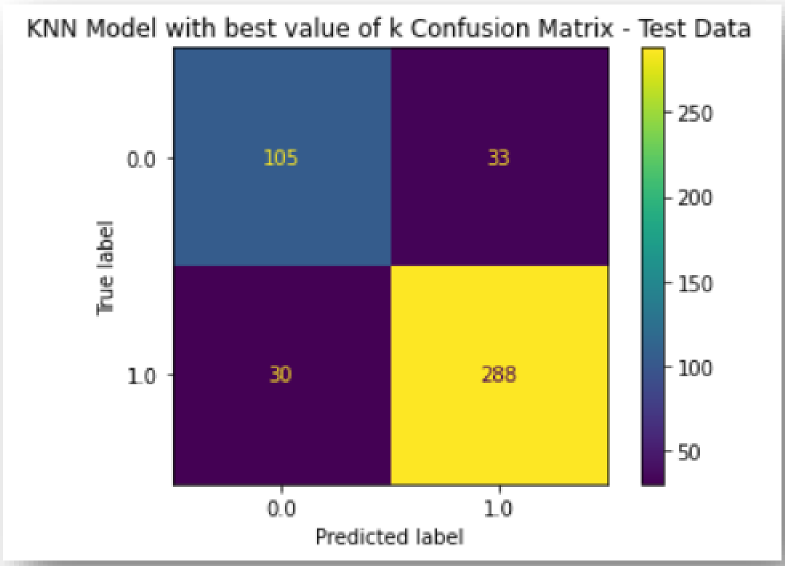


Figure 91: Confusion Matrix of the KNN Model with best value of k for Test Data

The confusion matrix was calculated and shown in Figure 91. The accuracy score for the test data was found to be 0.86 which is shown in Figure 90.

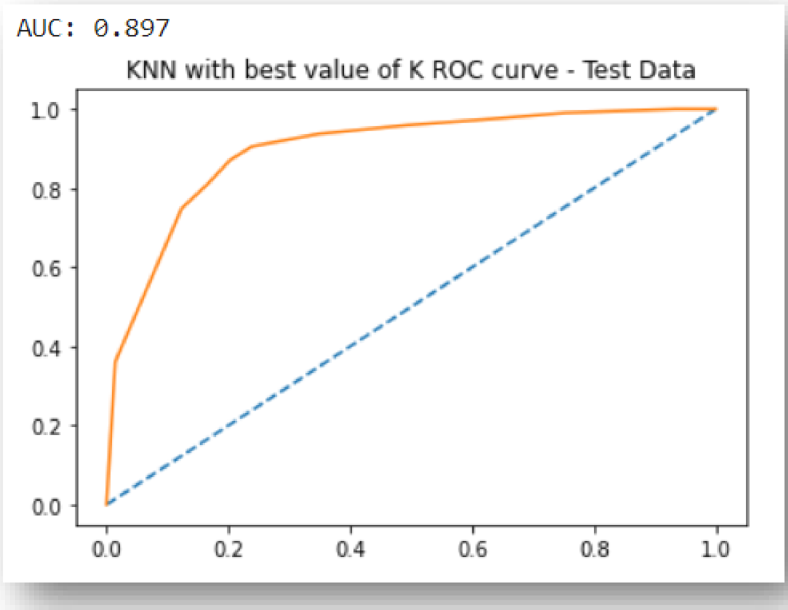
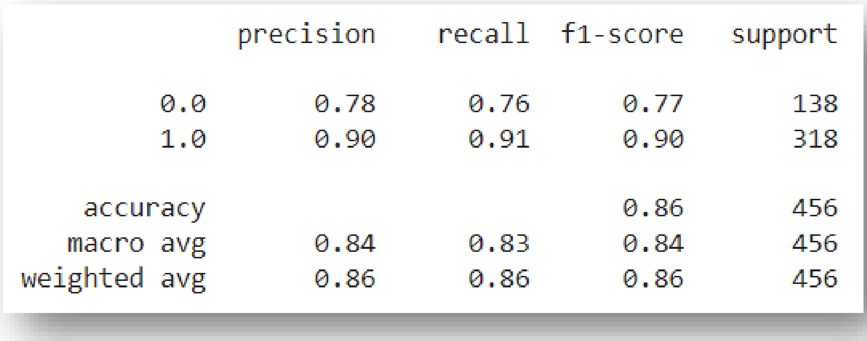


Figure 92: KNN Model with best value of K ROC Curve for Test Data

The ROC_AUC score for the test data was calculated to be 0.897. The ROC curve was plotted and is shown in Figure 92.



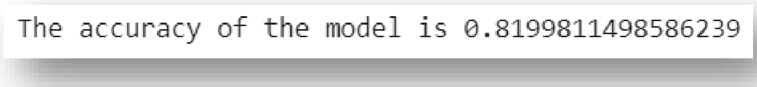
```
              precision    recall  f1-score   support

         0.0       0.78      0.76      0.77       138
         1.0       0.90      0.91      0.90       318

    accuracy                           0.86       456
   macro avg       0.84      0.83      0.84       456
weighted avg       0.86      0.86      0.86       456
```

Figure 93: KNN Model with the best value of K Classification Report for Test Data

The classification report for the KNN Model with the best value of K with the scores for the different model performance measures is calculated and shown in Figure 93. From this figure we can see that the precision of the model is 0.90 means 90% of the data points identified as positive by the model, are really positive. The f1-score is 0.90 means the model is 90% accurate on this data set. The model has 86% accuracy. The recall score is 0.91 which means 91% of the positive observations are correctly predicted.

Naïve Bayes Model

Training Data



```
The accuracy of the model is 0.8199811498586239
```

Figure 94: Accuracy of the NB Model for Training Data

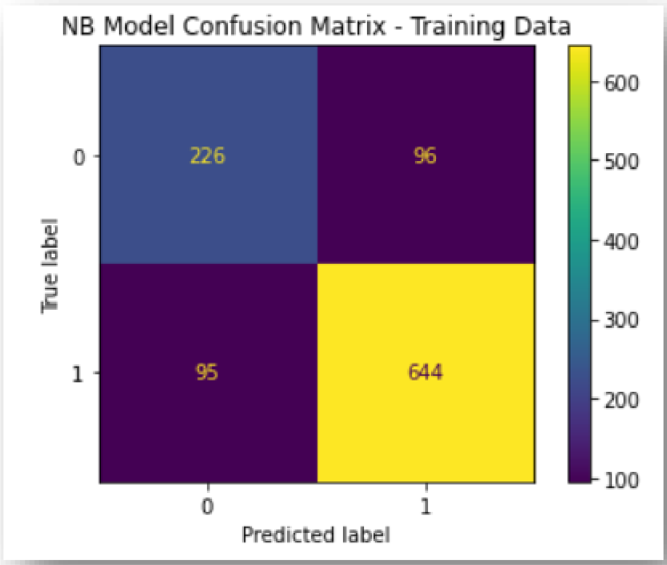


Figure 95: Confusion Matrix of the NB Model for Training Data

The confusion matrix was calculated and shown in Figure 95. The accuracy score for the training data was found to be 0.82 which is shown in Figure 94.

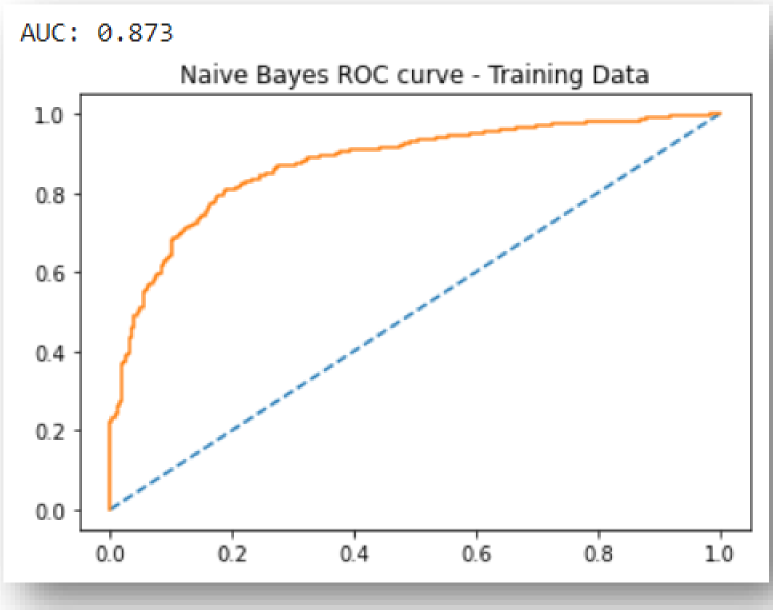
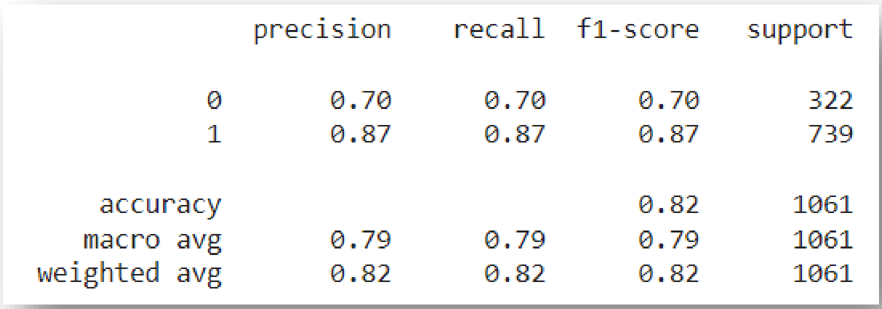


Figure 96: NB Model ROC Curve for Training Data

The ROC_AUC score for the training data was calculated to be 0.873. The ROC curve was plotted and is shown in Figure 96.

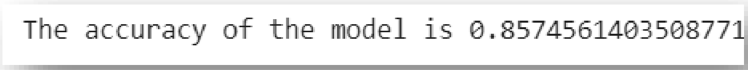


```
              precision    recall  f1-score   support

           0       0.70      0.70      0.70       322
           1       0.87      0.87      0.87       739

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

Figure 97: NB Model Classification Report for Training Data

The classification report for the Naïve Bayes Model with the scores for the different model performance measures is calculated and shown in Figure 97. From this figure we can see that the precision of the model is 0.87 means 87% of the data points identified as positive by the model, are really positive. The f1-score is 0.87 means the model is 87% accurate on this data set. The model has 82% accuracy. The recall score is 0.87 which means 87% of the positive observations are correctly predicted.

Test Data



```
The accuracy of the model is 0.8574561403508771
```

Figure 98: Accuracy of the NB Model for Test Data

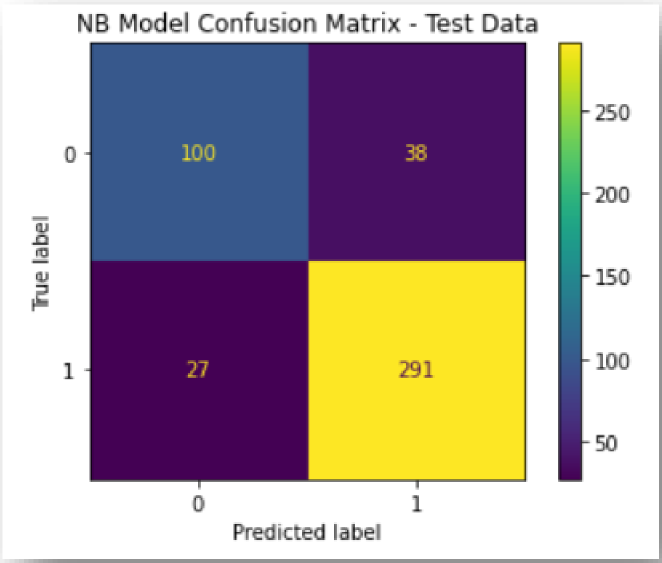


Figure 99: Confusion Matrix of the NB Model for Test Data

The confusion matrix was calculated and shown in Figure 99. The accuracy score for the test data was found to be 0.86 which is shown in Figure 98.

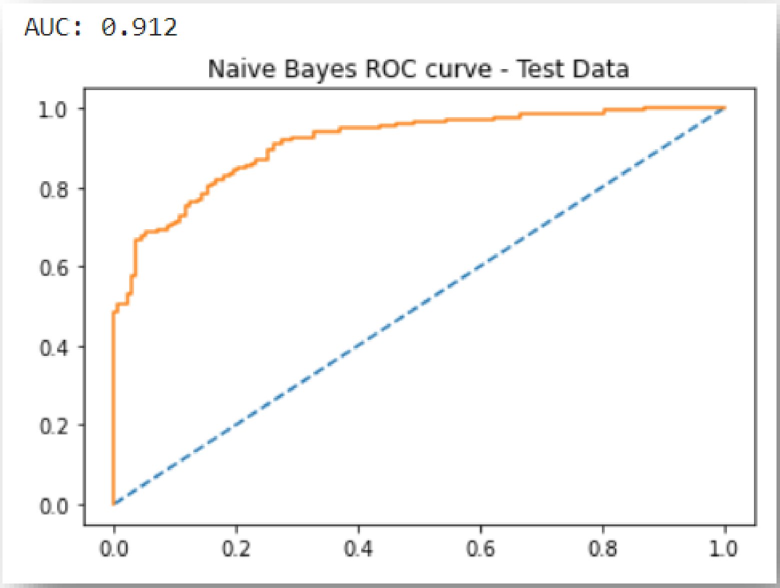
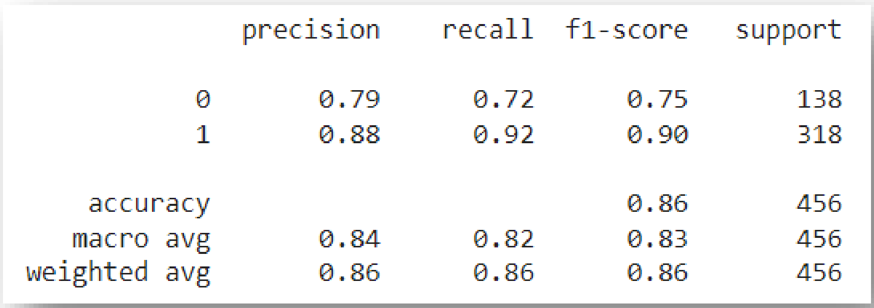


Figure 100: NB Model ROC Curve for Test Data

The ROC_AUC score for the test data was calculated to be 0.912. The ROC curve was plotted and is shown in Figure 100.



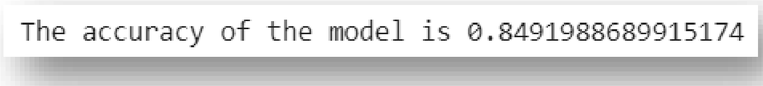
```
              precision    recall  f1-score   support

           0       0.79      0.72      0.75       138
           1       0.88      0.92      0.90       318

    accuracy                           0.86       456
   macro avg       0.84      0.82      0.83       456
weighted avg       0.86      0.86      0.86       456
```

Figure 101: NB Model Classification Report for Test Data

The classification report for the Naïve Bayes Model with the scores for the different model performance measures is calculated and shown in Figure 101. From this figure we can see that the precision of the model is 0.88 means 88% of the data points identified as positive by the model, are really positive. The f1-score is 0.90 means the model is 90% accurate on this data set. The model has 86% accuracy. The recall score is 0.92 which means 92% of the positive observations are correctly predicted.

Ada Boosting

Training Data



```
The accuracy of the model is 0.8491988689915174
```

Figure 102: Accuracy of Ada Boosting for Training Data

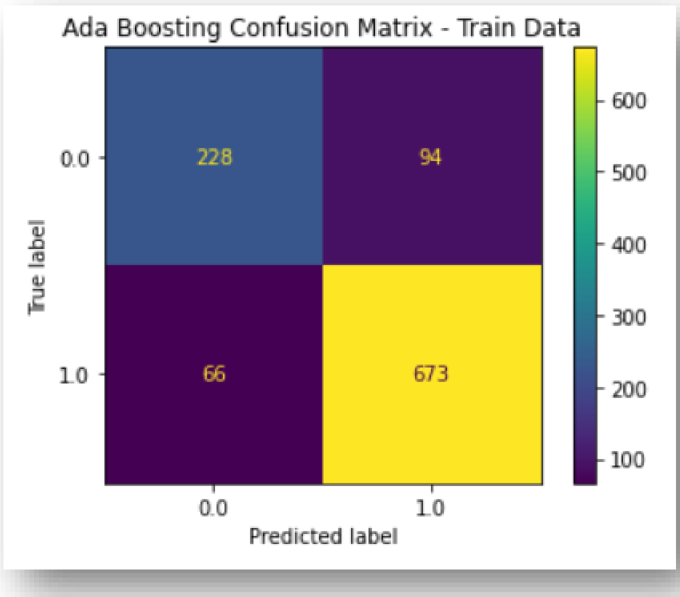


Figure 103: Confusion Matrix of Ada Boosting for Training Data

The confusion matrix was calculated and shown in Figure 103. The accuracy score for the training data was found to be 0.85 which is shown in Figure 102.

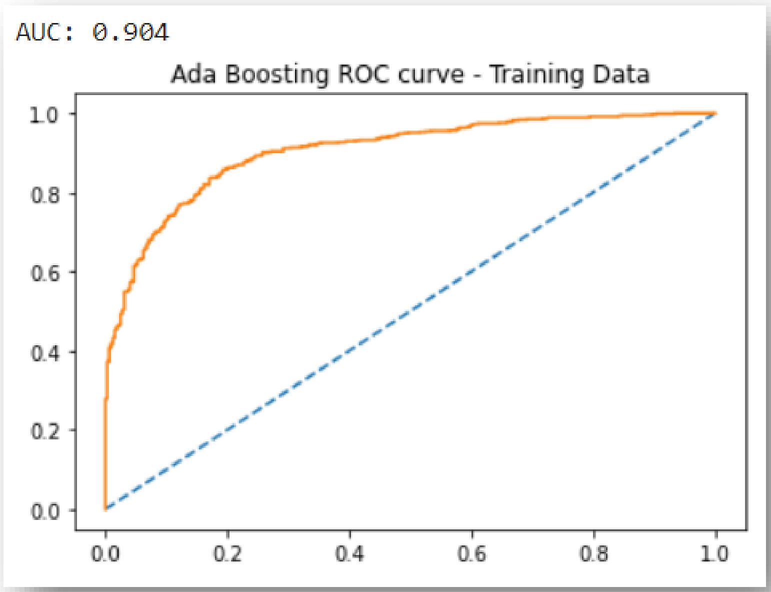


Figure 104: Ada Boosting ROC Curve for Training Data

The ROC_AUC score for the training data was calculated to be 0.904. The ROC curve was plotted and is shown in Figure 104.

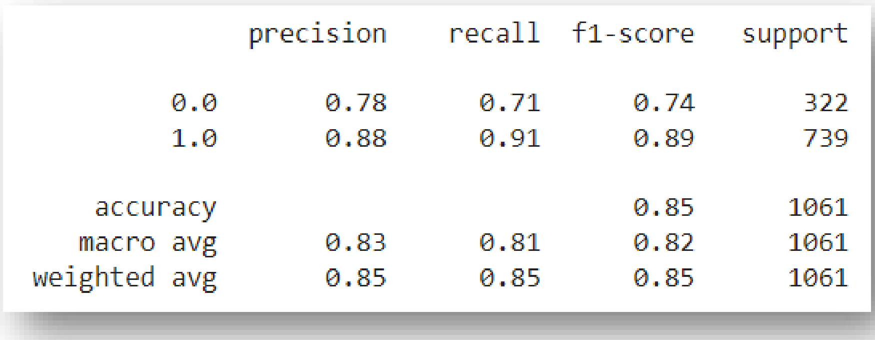
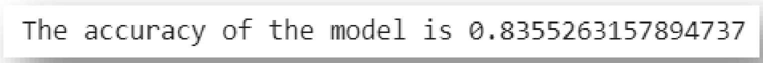

```
              precision    recall  f1-score   support

         0.0       0.78      0.71      0.74       322
         1.0       0.88      0.91      0.89       739

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

Figure 105: Ada Boosting Classification Report for Training Data

The classification report for the Ada Boosting Model with the scores for the different model performance measures is calculated and shown in Figure 105. From this figure we can see that the precision of the model is 0.88 means 88% of the data points identified as positive by the model, are really positive. The f1-score is 0.89 means the model is 89% accurate on this data set. The model has 85% accuracy. The recall score is 0.91 which means 91% of the positive observations are correctly predicted.

Test Data



```
The accuracy of the model is 0.8355263157894737
```

Figure 106: Accuracy of Ada Boosting for Test Data

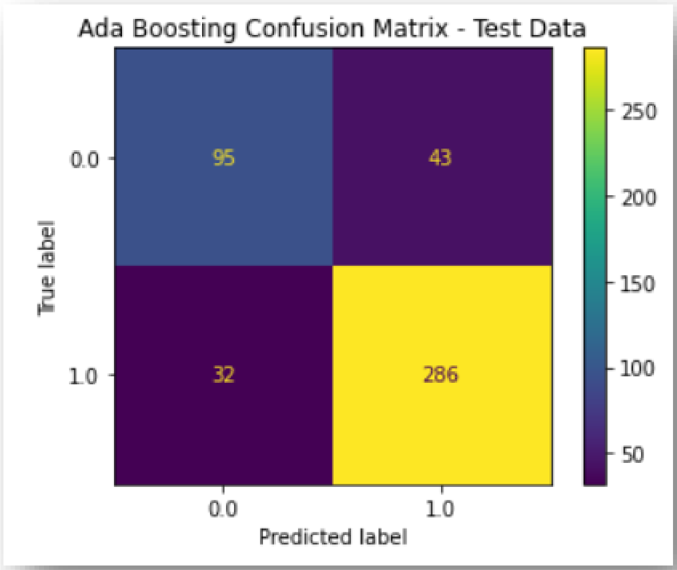


Figure 107: Confusion Matrix of Ada Boosting for Test Data

The confusion matrix was calculated and shown in Figure 107. The accuracy score for the test data was found to be 0.84 which is shown in Figure 106.

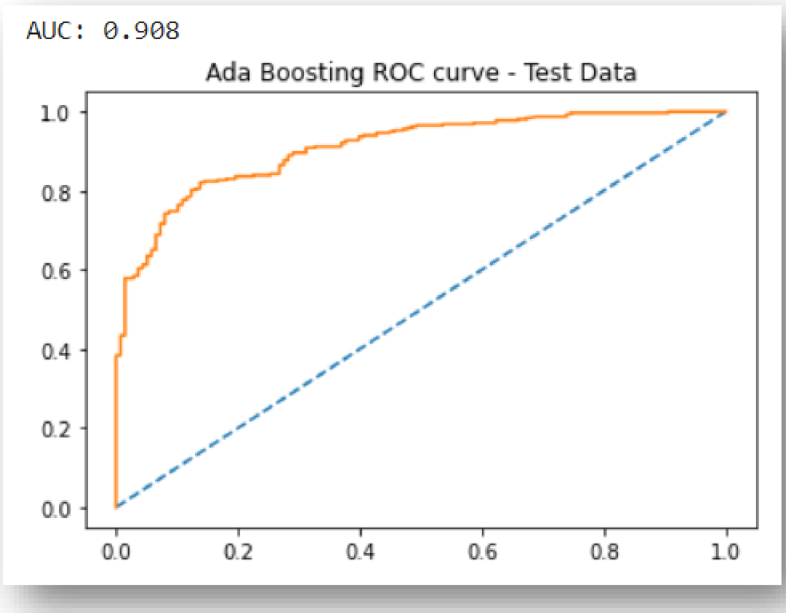
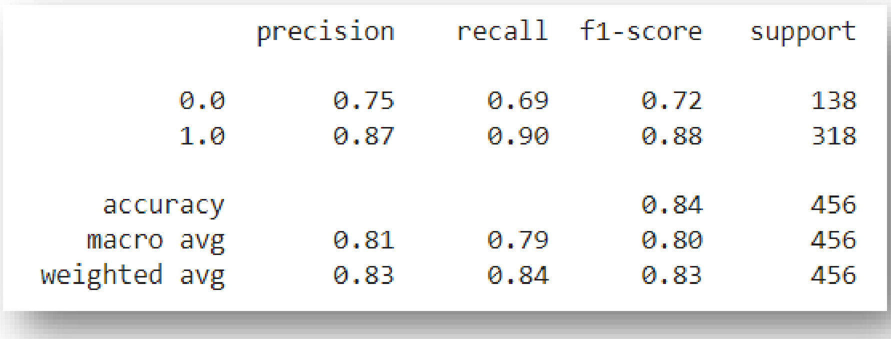


Figure 108: Ada Boosting ROC Curve for Test Data

The ROC_AUC score for the test data was calculated to be 0.908. The ROC curve was plotted and is shown in Figure 108.



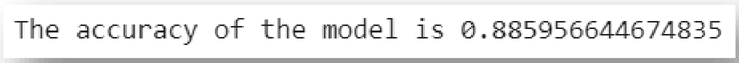
```
              precision    recall  f1-score   support

         0.0       0.75      0.69      0.72       138
         1.0       0.87      0.90      0.88       318

    accuracy                           0.84       456
   macro avg       0.81      0.79      0.80       456
weighted avg       0.83      0.84      0.83       456
```

Figure 109: Ada Boosting Classification Report for Test Data

The classification report for the Ada Boosting Model with the scores for the different model performance measures is calculated and shown in Figure 109. From this figure we can see that the precision of the model is 0.87 means 87% of the data points identified as positive by the model, are really positive. The f1-score is 0.88 means the model is 88% accurate on this data set. The model has 84% accuracy. The recall score is 0.90 which means 90% of the positive observations are correctly predicted.

Gradient Boosting

Training Data



```
The accuracy of the model is 0.885956644674835
```

Figure 110: Accuracy of Gradient Boosting for Training Data

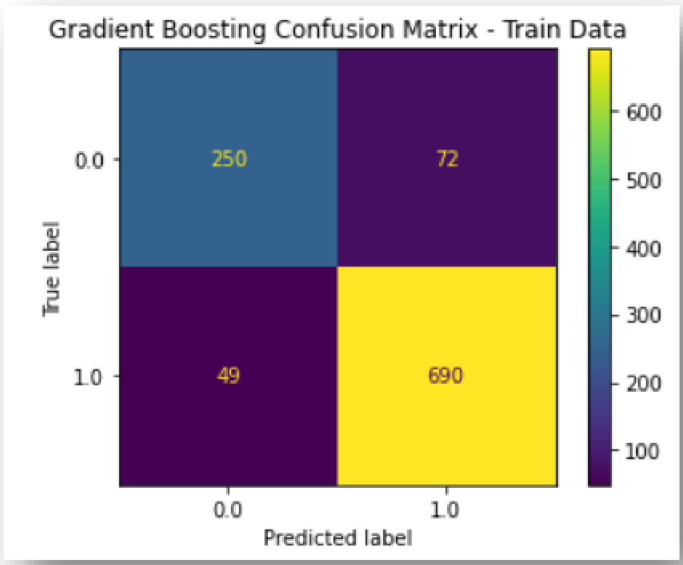


Figure 111: Confusion Matrix of Gradient Boosting for Training Data

The confusion matrix was calculated and shown in Figure 111. The accuracy score for the training data was found to be 0.89 which is shown in Figure 110.

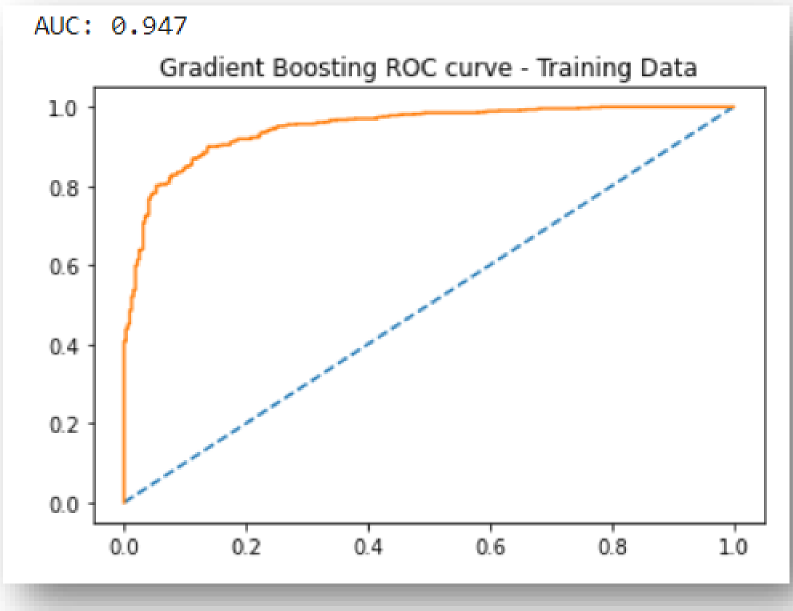
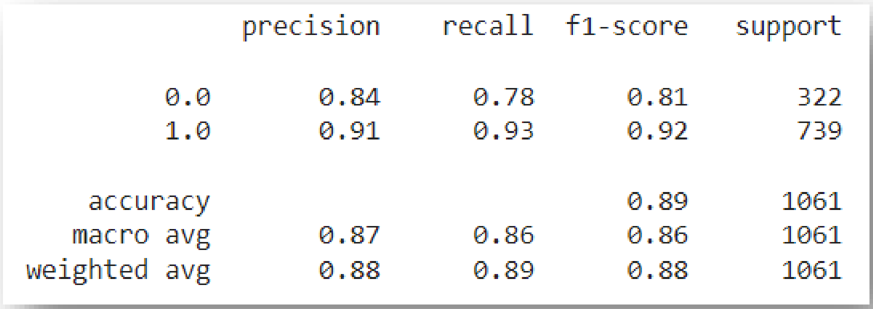


Figure 112: Gradient Boosting ROC Curve for Training Data

The ROC_AUC score for the training data was calculated to be 0.947. The ROC curve was plotted and is shown in Figure 112.

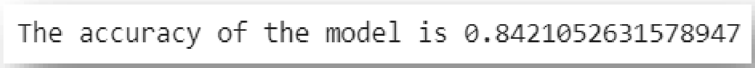


```
              precision    recall  f1-score   support

         0.0       0.84      0.78      0.81       322
         1.0       0.91      0.93      0.92       739

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

Figure 113: Gradient Boosting Classification Report for Training Data

The classification report for the Gradient Boosting Model with the scores for the different model performance measures is calculated and shown in Figure 113. From this figure we can see that the precision of the model is 0.91 means 91% of the data points identified as positive by the model, are really positive. The f1-score is 0.92 means the model is 92% accurate on this data set. The model has 89% accuracy. The recall score is 0.93 which means 93% of the positive observations are correctly predicted.

Test Data



```
The accuracy of the model is 0.8421052631578947
```

Figure 114: Accuracy of Gradient Boosting for Test Data

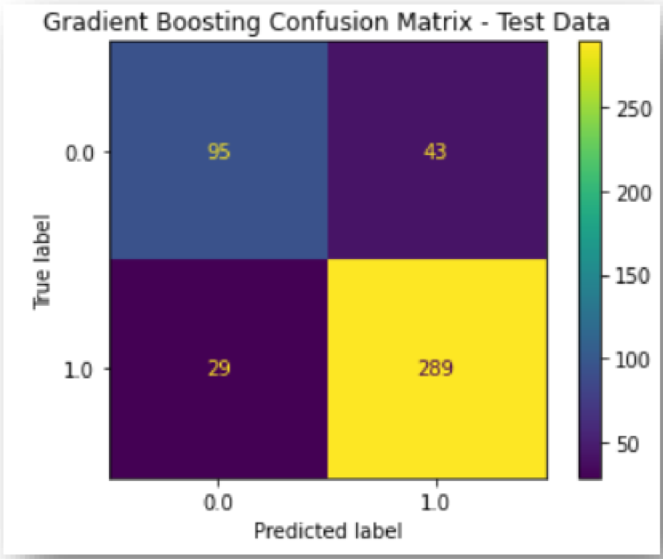


Figure 115: Confusion Matrix of Gradient Boosting for Test Data

The confusion matrix was calculated and shown in Figure 115. The accuracy score for the test data was found to be 0.84 which is shown in Figure 114.

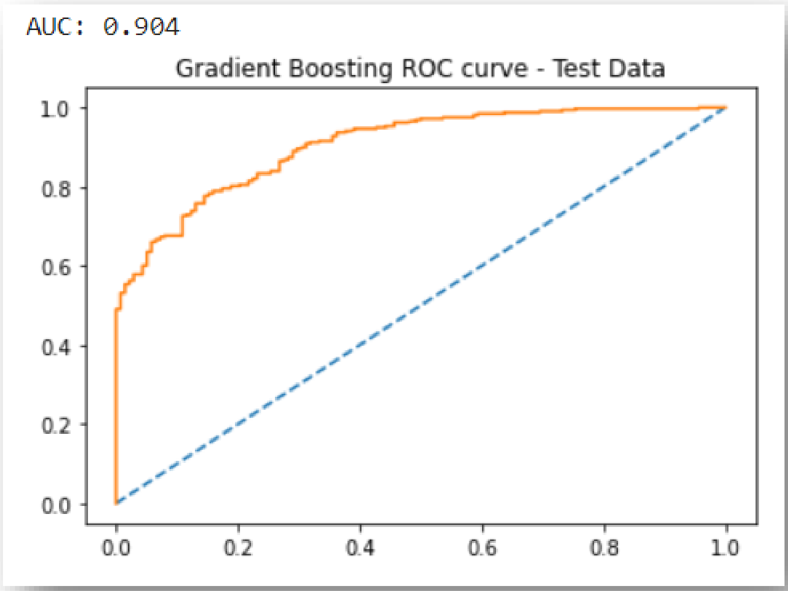
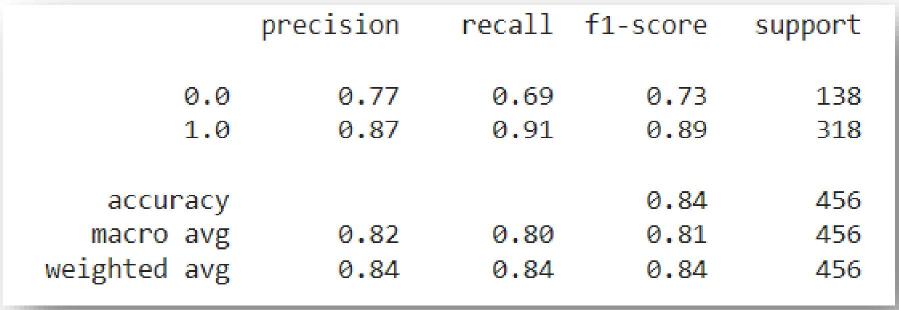


Figure 116: Gradient Boosting ROC Curve for Test Data

The ROC_AUC score for the test data was calculated to be 0.904. The ROC curve was plotted and is shown in Figure 116.



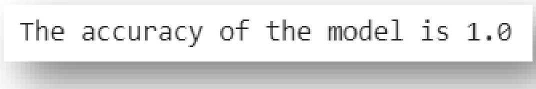
```
              precision    recall  f1-score   support

         0.0       0.77      0.69      0.73       138
         1.0       0.87      0.91      0.89       318

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

Figure 117: Gradient Boosting Classification Report for Test Data

The classification report for the Gradient Boosting Model with the scores for the different model performance measures is calculated and shown in Figure 117. From this figure we can see that the precision of the model is 0.87 means 87% of the data points identified as positive by the model, are really positive. The f1-score is 0.89 means the model is 89% accurate on this data set. The model has 84% accuracy. The recall score is 0.91 which means 91% of the positive observations are correctly predicted.

Random Forest

Training Data



```
The accuracy of the model is 1.0
```

Figure 118: Accuracy of Random Forest for Training Data

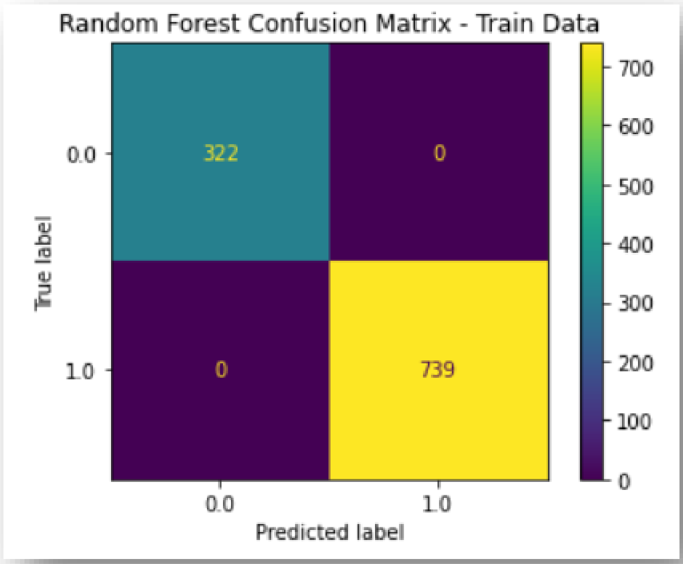


Figure 119: Confusion Matrix of Random Forest for Training Data

The confusion matrix was calculated and shown in Figure 119. The accuracy score for the training data was found to be 1.0 which is shown in Figure 118.

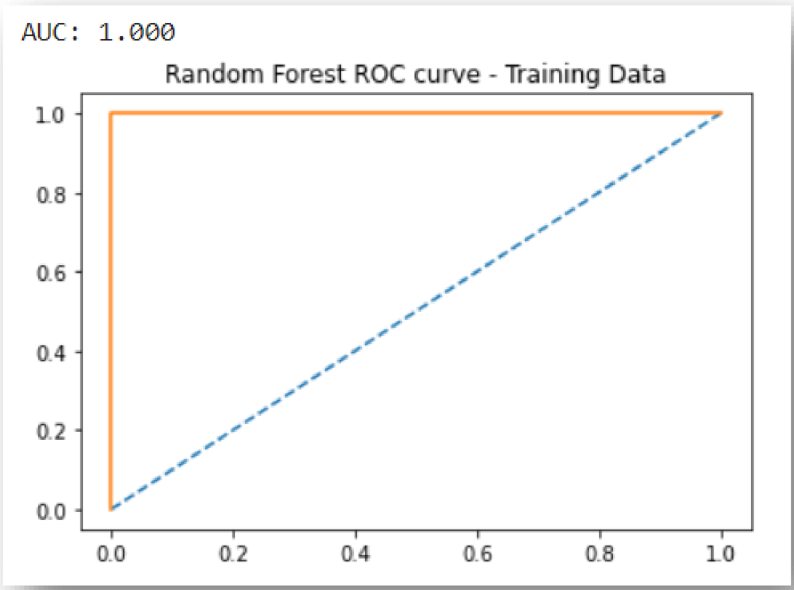
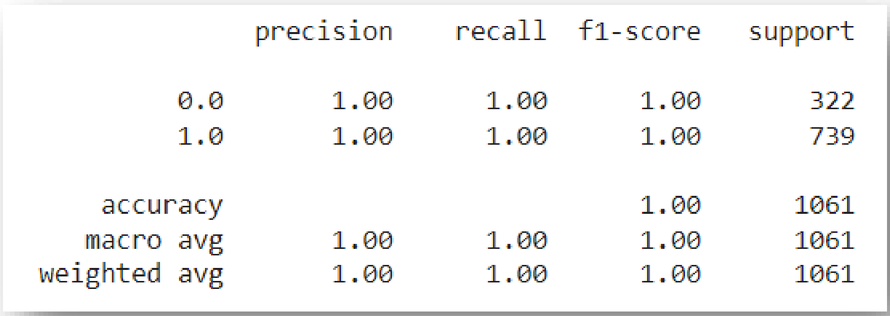


Figure 120: Random Forest ROC Curve for Training Data

The ROC_AUC score for the training data was calculated to be 1.000. The ROC curve was plotted and is shown in Figure 120.

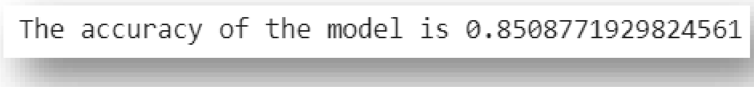


```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       322
         1.0       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
```

Figure 121: Random Forest Classification Report for Training Data

The classification report for the Random Forest Model with the scores for the different model performance measures is calculated and shown in Figure 121. From this figure we can see that the precision of the model is 1.00 means 100% of the data points identified as positive by the model, are really positive. The f1-score is 1.00 means the model is 100% accurate on this data set. The model has 100% accuracy. The recall score is 1.00 which means 100% of the positive observations are correctly predicted.

Test Data



```
The accuracy of the model is 0.8508771929824561
```

Figure 122: Accuracy of Random Forest for Test Data

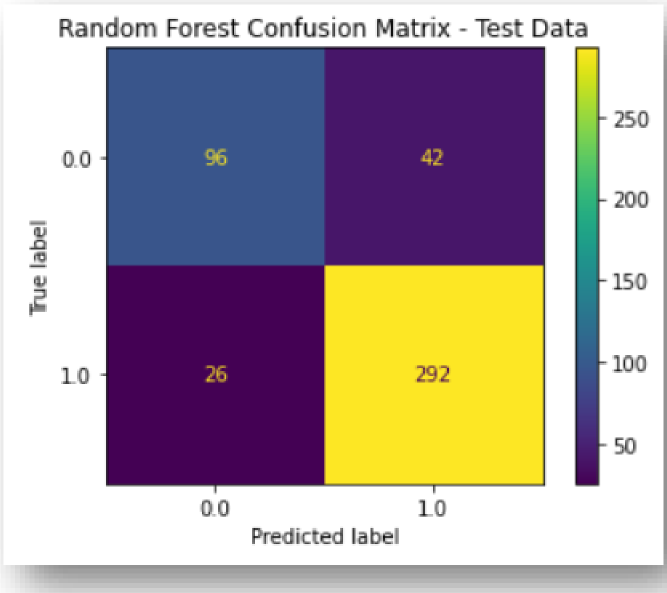


Figure 123: Confusion Matrix of Random Forest for Test Data

The confusion matrix was calculated and shown in Figure 123. The accuracy score for the test data was found to be 0.85 which is shown in Figure 122.

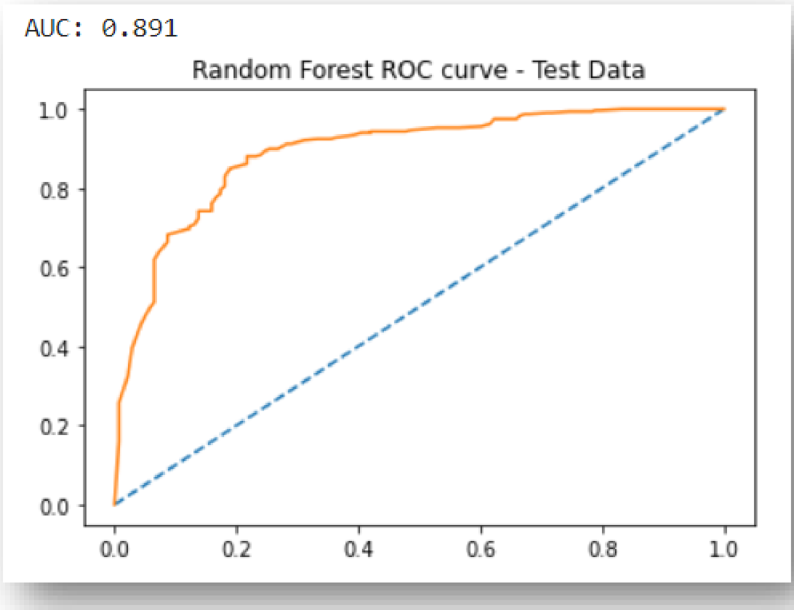

```
AUC: 0.891
```

Figure 124: Random Forest ROC Curve for Test Data

The ROC_AUC score for the test data was calculated to be 0.891. The ROC curve was plotted and is shown in Figure 124.

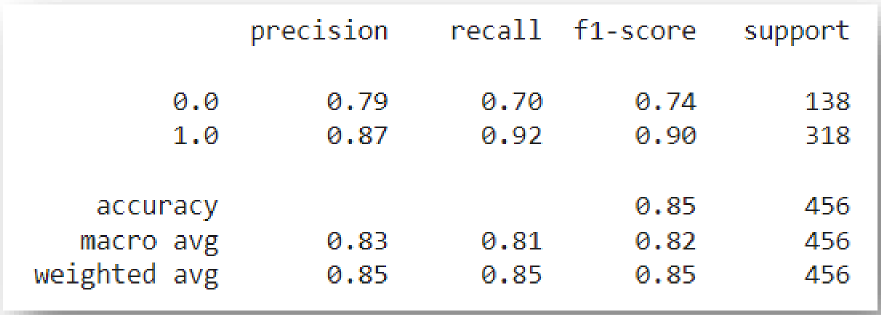
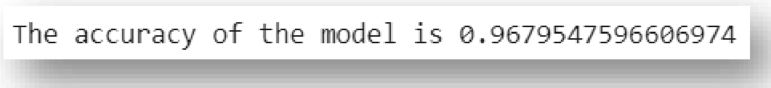

```
              precision    recall  f1-score   support

         0.0       0.79      0.70      0.74       138
         1.0       0.87      0.92      0.90       318

    accuracy                           0.85       456
   macro avg       0.83      0.81      0.82       456
weighted avg       0.85      0.85      0.85       456
```

Figure 125: Random Forest Classification Report for Test Data

The classification report for the Random Forest Model with the scores for the different model performance measures is calculated and shown in Figure 125. From this figure we can see that the precision of the model is 0.87 means 87% of the data points identified as positive by the model, are really positive. The f1-score is 0.90 means the model is 90% accurate on this data set. The model has 85% accuracy. The recall score is 0.92 which means 92% of the positive observations are correctly predicted.

Bagging Model (Using Random Forest Classifier)

Training Data



```
The accuracy of the model is 0.9679547596606974
```

Figure 126: Accuracy of Bagging for Training Data

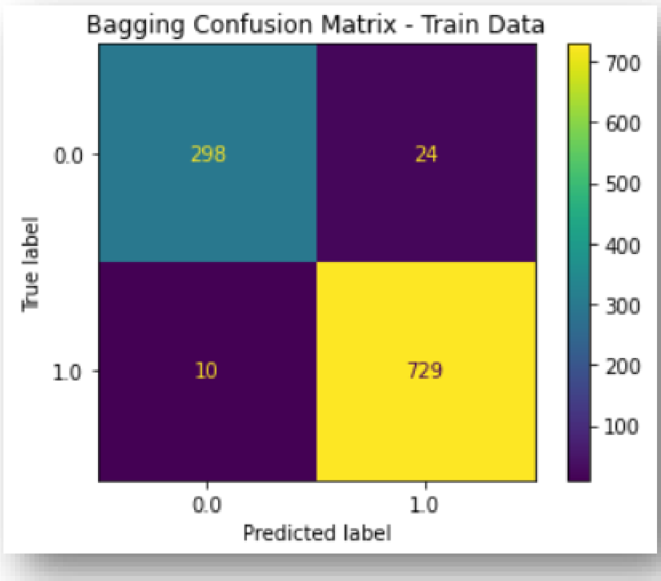


Figure 127: Confusion Matrix of Bagging for Training Data

The confusion matrix was calculated and shown in Figure 127. The accuracy score for the training data was found to be 0.97 which is shown in Figure 126.

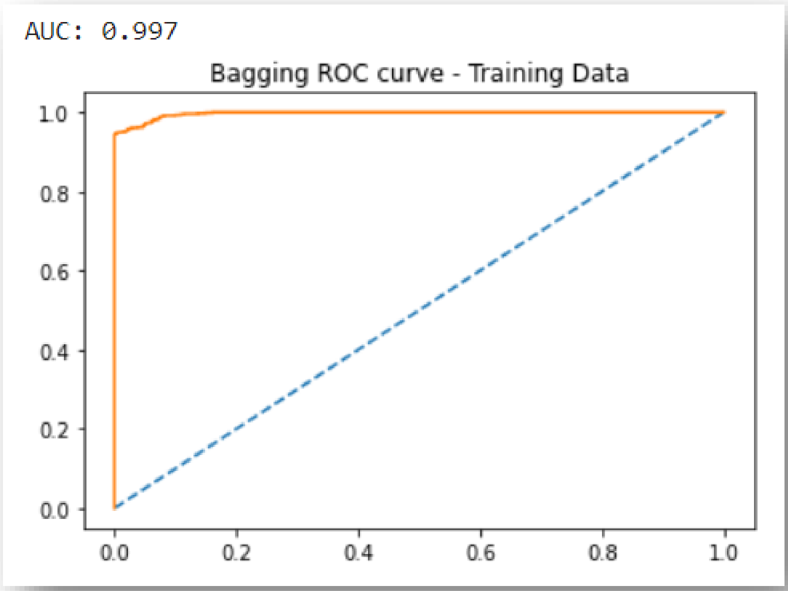
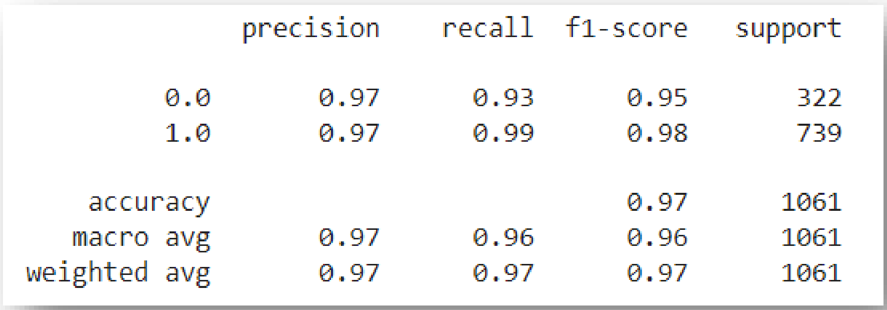


Figure 128: Bagging ROC Curve for Training Data

The ROC_AUC score for the training data was calculated to be 0.997. The ROC curve was plotted and is shown in Figure 128.

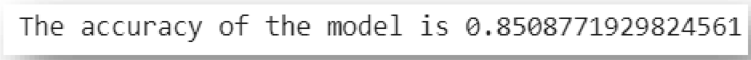


```
              precision    recall  f1-score   support

         0.0       0.97      0.93      0.95       322
         1.0       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
```

Figure 129: Bagging Classification Report for Training Data

The classification report for the Bagging Model with the scores for the different model performance measures is calculated and shown in Figure 129. From this figure we can see that the precision of the model is 0.97 means 97% of the data points identified as positive by the model, are really positive. The f1-score is 0.98 means the model is 98% accurate on this data set. The model has 97% accuracy. The recall score is 0.99 which means 99% of the positive observations are correctly predicted.

Test Data



```
The accuracy of the model is 0.8508771929824561
```

Figure 130: Accuracy of Bagging for Test Data

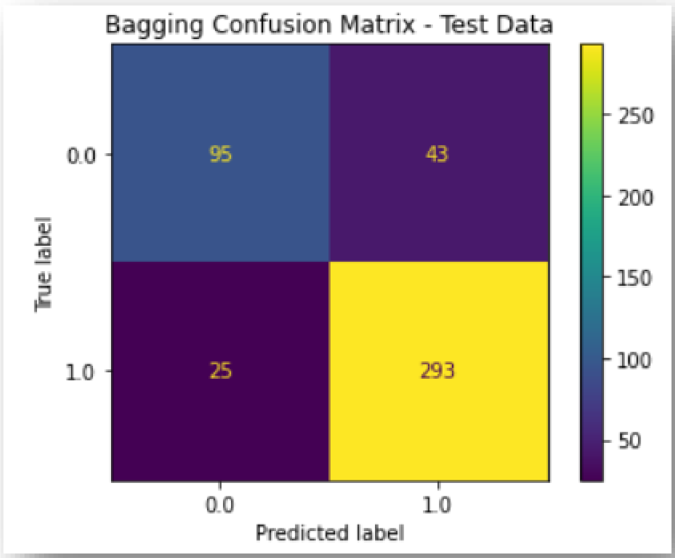


Figure 131: Confusion Matrix of Bagging for Test Data

The confusion matrix was calculated and shown in Figure 131. The accuracy score for the test data was found to be 0.85 which is shown in Figure 130.

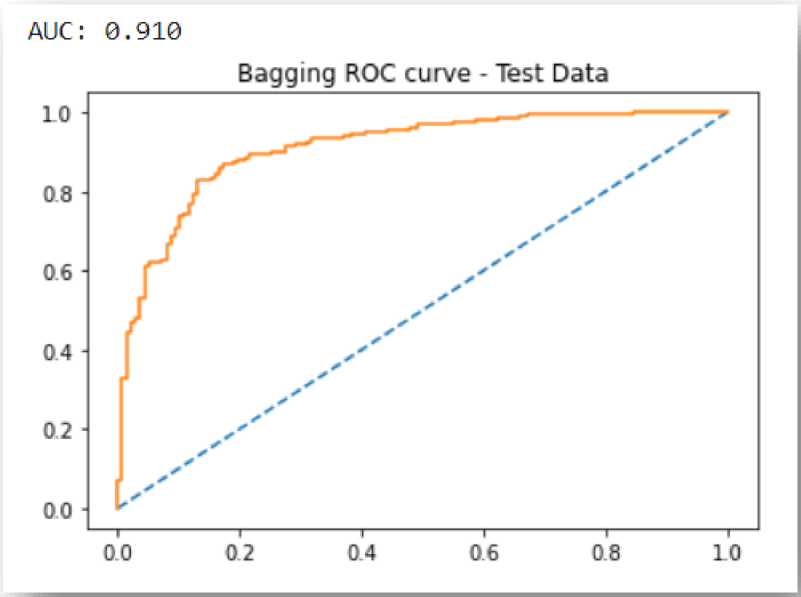
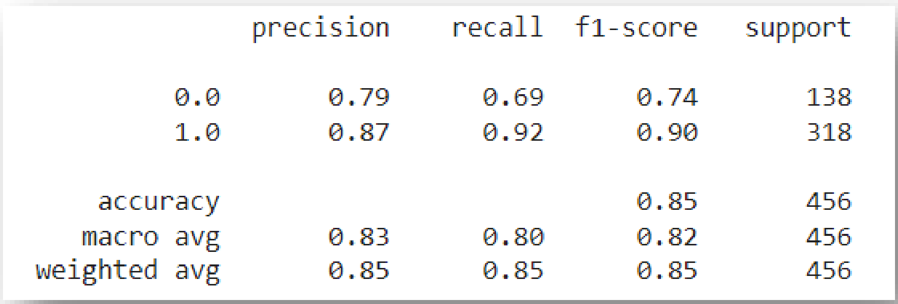


Figure 132: Bagging ROC Curve for Test Data

The ROC_AUC score for the test data was calculated to be 0.910. The ROC curve was plotted and is shown in Figure 132.



```
              precision    recall  f1-score   support

         0.0       0.79      0.69      0.74       138
         1.0       0.87      0.92      0.90       318

    accuracy                           0.85       456
   macro avg       0.83      0.80      0.82       456
weighted avg       0.85      0.85      0.85       456
```

Figure 133: Bagging Classification Report for Test Data

The classification report for the Bagging Model with the scores for the different model performance measures is calculated and shown in Figure 133. From this figure we can see that the precision of the model is 0.87 means 87% of the data points identified as positive by the model, are really positive. The f1-score is 0.90 means the model is 90% accurate on this data set. The model has 85% accuracy. The recall score is 0.92 which means 92% of the positive observations are correctly predicted.

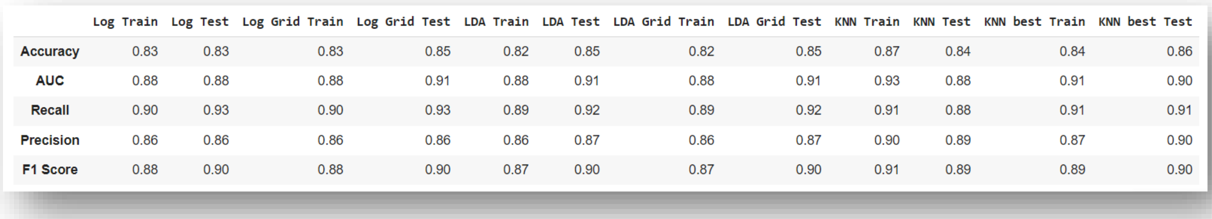
Model Comparision:

A comparison of all the models was done to understand which model is the best suited for our case study. There are a total of 11 different models. The basis on which the models are evaluated are known as performance metrics. The metrics on which the model will be evaluated are

- Accuracy
- AUC
- Recall
- Precision
- F1 Score

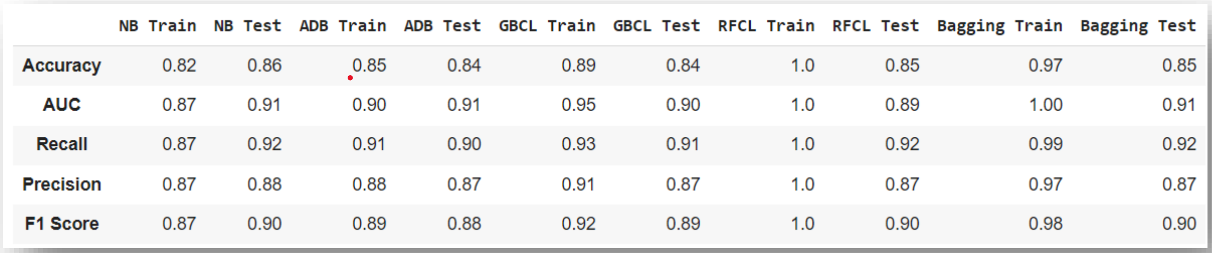
The model performance measures of all the models were tabulated and it is shown in Table 34 and Table 35.

Table 34: Comparison of all the models - 1



	Log Train	Log Test	Log Grid Train	Log Grid Test	LDA Train	LDA Test	LDA Grid Train	LDA Grid Test	KNN Train	KNN Test	KNN best Train	KNN best Test
Accuracy	0.83	0.83	0.83	0.85	0.82	0.85	0.82	0.85	0.87	0.84	0.84	0.86
AUC	0.88	0.88	0.88	0.91	0.88	0.91	0.88	0.91	0.93	0.88	0.91	0.90
Recall	0.90	0.93	0.90	0.93	0.89	0.92	0.89	0.92	0.91	0.88	0.91	0.91
Precision	0.86	0.86	0.86	0.86	0.86	0.87	0.86	0.87	0.90	0.89	0.87	0.90
F1 Score	0.88	0.90	0.88	0.90	0.87	0.90	0.87	0.90	0.91	0.89	0.89	0.90

Table 35: Comparison of all the models - 2



	NB Train	NB Test	ADB Train	ADB Test	GBCL Train	GBCL Test	RFCL Train	RFCL Test	Bagging Train	Bagging Test
Accuracy	0.82	0.86	0.85	0.84	0.89	0.84	1.0	0.85	0.97	0.85
AUC	0.87	0.91	0.90	0.91	0.95	0.90	1.0	0.89	1.00	0.91
Recall	0.87	0.92	0.91	0.90	0.93	0.91	1.0	0.92	0.99	0.92
Precision	0.87	0.88	0.88	0.87	0.91	0.87	1.0	0.87	0.97	0.87
F1 Score	0.87	0.90	0.89	0.88	0.92	0.89	1.0	0.90	0.98	0.90

From Table 34 and Table 35, on the basis of

- Accuracy – KNN with K value 11 and Naïve Bayes model performed better than the others
- AUC – all the models except random forest and bagging performed better
- Recall – Logistic Regression and Logistic Regression using Grid Search CV performed equally well
- Precision – KNN model, KNN model with the best value of K and Naïve Bayes model performed better than other models
- F1 Score – Linear Discriminant Analysis and Naïve Bayes model performed better than other models

On the basis of accuracy, KNN model with the best value of K and Naïve Bayes model performed better than others. The Naïve Bayes model test scores for other the performance metrics are better compared to other models. KNN model has some disadvantages.

Disadvantages of KNN Model:

- It does not output any models. It calculates distances for every new point (lazy learner)
- It is computationally intensive
- Accuracy depends on the quality of the data
- With large data, the prediction stage might be slow
- Sensitive to the scale of the data and irrelevant features

Due to these disadvantages, **the Naïve Bayes model is chosen as the best and optimized model.**

Inference: 5 marks

1.8 Based on these predictions, what are the insights? (5 marks)

Observations from the analysis of data

- Labour party is performing better than the conservative party and there is a huge difference in the margins.
- Those who have better national economic conditions are preferring to vote for labour party.
- Female voters are the majority of voters than the turn out is greater than the males.
- People having higher Eurosceptic sentiments conservative party are preferring to vote for conservative party.
- Those who have higher political knowledge have voted for conservative party.
- Looking at the assessment for both the leaders, Labour leader is performing well as he has got better ratings in the assessment.

Insights and Recommendations

The main objective of this problem is to build a model to predict for which party a voter will vote for on the basis of the given information and to create an exit poll that will help in predicting the overall win and seats covered by a particular party.

The model was built on a dataset of approximately 1500 values meaning the survey was conducted on 1500 people. However, the population of the entire state will be much higher than 1500. Therefore, when we try to predict the data with the survey from the entire state, we will be able to predict which party will win the elections.

Once we predict which party will win the elections, talk shows and programs can be arranged with the future winning party.

We can use this prediction results and form business dealing with the winning party. We can help them win the elections while the future winning party can offer us press rights or appoint us as their official analyst.

The best model chosen for this data is Naïve Bayes model and this model predicts which party a voter will vote for with 86% accuracy. Once the survey is conducted on the entire population and the predictions with the model is made, we will get an idea of which party will win the elections.

From the predictions we will get an idea of which party has more chance of winning in a particular constituency. If a party has moderate or less chances of winning in a particular constituency, more campaigns can be conducted in that area to increase the chances of winning.

If a party has high chances of winning in a particular constituency, those areas must not be left out carelessly as the opposition party will be concentrating on these areas to increase their chances of winning. The party has to continue campaigning in these regions.

From analysis, it is seen that people with higher political knowledge and Eurosceptic sentiments tend to vote for the conservative party. Therefore, conservative party leaders can arrange meetings and talks with educated and politically knowledgeable people to increase their winning chances.

The common problems of people can be studied and the solution to their problems can be included in the election manifesto. By doing this, people will more likely vote for the party which offers solution to their problems and hence increase the chances of winning for that particular party.