Machine Learning Business Report

Rohan R. Khade

Table of Contents

Chapter 1. Problem 1 .... - 7 -

1.1 Problem Statement .... - 7 -

1.1.1 Data Dictionary: .... - 7 -

1.1.2 Read the dataset. Do the descriptive statistics and do the null value condition check. Write an inference on it. .... - 7 -

1.1.3 Perform Univariate and Bivariate Analysis. Do exploratory data analysis. Check for Outliers. .... - 10 -

1.1.4 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). .... - 21 -

1.1.5 Apply Logistic Regression and LDA (linear discriminant analysis). .... - 22 -

1.1.5.1 Logistic Regression .... - 22 -

1.1.5.2 Linear Discriminant Analysis (LDA) .... - 25 -

1.1.6 Apply KNN Model and Naïve Bayes Model. Interpret the results. .... - 27 -

1.1.6.1 KNN .... - 27 -

1.1.6.2 Naïve Bayes .... - 30 -

1.1.7 Model Tuning, Bagging (Random Forest should be applied for Bagging), and Boosting. .... - 32 -

1.1.8 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. .... - 32 -

1.1.8.1 Bagging (on Random Forest Classifier) .... - 32 -

1.1.8.2 ADA Boosting .... - 34 -

1.1.8.3 Gradient Boosting .... - 37 -

1.1.9 Model Tuning with Hyper-parameters .... - 40 -

1.1.9.1 Logistic Regression .... - 40 -

1.1.9.2 LDA .... - 42 -

1.1.9.3 Naive Bayes .... - 45 -

1.1.9.4 KNN .... - 47 -

1.1.10 Best Model: .... - 49 -

1.1.11 Based on these predictions, what are the insights? .... - 50 -

Chapter 2. Problem 2 .... - 51 -

2.1 Problem Statement .... - 51 -

2.1.1 Find the number of characters, words, and sentences for the mentioned documents. .... - 51 -

2.1.2 Remove all the stopwords from all three speeches. .... - 51 -

Back to Top

2.1.3 Which word occurs the most number of times in his inaugural address for each president? Mention the top three words. (after removing the stopwords) - 52 -
2.1.4 Plot the word cloud of each of the speeches of the variable. (after removing the stopwords) - 53 -

List of Tables

Table 1 Dataframe: df (with head function) - 7 -
Table 2 Dataframe: df (with describe function) - 8 -
Table 3 Duplicate entries in df - 9 -
Table 4 Combined Model: Labour - 40 -
Table 5 Combined Model: Conservative - 40 -

List of Figures

Figure 1. Dataset information - 8 -
Figure 2. Skewness - 10 -
Figure 3. National economic condition data series: Description & graphical representation - 10 -
Figure 4. Household economic condition data series: Description & graphical representation - 11 -
Figure 5. Blair data series: Description & graphical representation - 12 -
Figure 6. Hague data series: Description & graphical representation - 13 -
Figure 7. Europe data series: Description & graphical representation - 14 -
Figure 8. Political knowledge data series: Description & graphical representation - 15 -
Figure 9. Pie chart analysis of the three variables (vote, votes for blair, votes for hague) - 16 -
Figure 10. Strip chart analysis of blair and age - 16 -
Figure 11. Strip chart analysis of hague and age - 17 -
Figure 12. Correlation matrix - 18 -
Figure 13. Pairplot - 19 -
Figure 14. Boxplot (with outliers) - 20 -
Figure 15. Boxplot (after outlier treatment) - 21 -
Figure 16. New Dataframe: df1 (with dummies) - 21 -
Figure 17. Predicted Classes and Probabilities - 22 -
Figure 18. Logistic Regression: Train AUC Curve - 23 -
Figure 19. Logistic Regression: Test AUC Curve - 23 -

Back to Top

Figure 20. Logistic Regression: Train Confusion Matrix - 24 -
Figure 21. Logistic Regression: Train Classification Report - 24 -
Figure 22. Logistic Regression: Test Confusion Matrix - 24 -
Figure 23. Logistic Regression: Test Classification Report - 24 -
Figure 24. Predicted Classes and Probabilities - 25 -
Figure 25. Linear Discriminant Analysis (LDA): Train AUC Curve - 25 -
Figure 26. Linear Discriminant Analysis (LDA): Test AUC Curve - 26 -
Figure 27. Linear Discriminant Analysis (LDA): Train Confusion Matrix - 26 -
Figure 28. Linear Discriminant Analysis (LDA): Train Classification Report - 26 -
Figure 29. Linear Discriminant Analysis (LDA): Test Confusion Matrix - 26 -
Figure 30. Linear Discriminant Analysis (LDA): Test Classification Report - 27 -
Figure 31. Predicted Classes and Probabilities - 27 -
Figure 32. KNN: Train AUC Curve - 28 -
Figure 33. KNN: Test AUC Curve - 28 -
Figure 34. KNN: Train Confusion Matrix - 29 -
Figure 35. KNN: Train Classification Report - 29 -
Figure 36. KNN: Test Confusion Matrix - 29 -
Figure 37. KNN: Test Classification Report - 29 -
Figure 38. Predicted Classes and Probabilities - 30 -
Figure 39. NB: Train AUC Curve - 30 -
Figure 40. NB: Test AUC Curve - 31 -
Figure 41. NB: Train Confusion Matrix - 31 -
Figure 42. NB: Train Classification Report - 31 -
Figure 43. NB: Test Confusion Matrix - 31 -
Figure 44. NB: Test Classification Report - 32 -
Figure 45. Predicted Classes and Probabilities - 32 -
Figure 46. Bagging: Train AUC Curve - 33 -
Figure 47. Bagging: Test AUC Curve - 33 -
Figure 48. Bagging: Train Confusion Matrix - 34 -
Figure 49. Bagging: Train Classification Report - 34 -
Figure 50. Bagging: Test Confusion Matrix - 34 -
Figure 51. Bagging: Test Classification Report - 34 -

Back to Top

Figure 52. Predicted Classes and Probabilities - 35 -
Figure 53. ADA Boosting: Train AUC Curve - 35 -
Figure 54. ADA Boosting: Test AUC Curve - 36 -
Figure 55. ADA Boosting: Train Confusion Matrix - 36 -
Figure 56. ADA Boosting: Train Classification Report - 36 -
Figure 57. ADA Boosting: Test Confusion Matrix - 36 -
Figure 58. ADA Boosting: Test Classification Report - 37 -
Figure 59. Predicted Classes and Probabilities - 37 -
Figure 60. Gradient Boosting: Train AUC Curve - 38 -
Figure 61. Gradient Boosting: Test AUC Curve - 38 -
Figure 62. Gradient Boosting: Train Confusion Matrix - 39 -
Figure 63. Gradient Boosting: Train Classification Report - 39 -
Figure 64. Gradient Boosting: Test Confusion Matrix - 39 -
Figure 65. Gradient Boosting: Test Classification Report - 39 -
Figure 66. Logistic Regression (Post Tuning): Train AUC Curve - 41 -
Figure 67. Logistic Regression (Post Tuning): Test AUC Curve - 41 -
Figure 68. Logistic Regression (Post Tuning): Train Confusion Matrix - 42 -
Figure 69. Logistic Regression (Post Tuning): Train Classification Report - 42 -
Figure 70. Logistic Regression (Post Tuning): Test Confusion Matrix - 42 -
Figure 71. Logistic Regression (Post Tuning): Test Classification Report - 42 -
Figure 72. LDA (Post Tuning): Train AUC Curve - 43 -
Figure 73. LDA (Post Tuning): Test AUC Curve - 43 -
Figure 74. LDA (Post Tuning): Train Confusion Matrix - 44 -
Figure 75. LDA (Post Tuning): Train Classification Report - 44 -
Figure 76. LDA (Post Tuning): Test Confusion Matrix - 44 -
Figure 77. LDA (Post Tuning): Test Classification Report - 44 -
Figure 78. Naive Bayes (Post Tuning): Train AUC Curve - 45 -
Figure 79. Naive Bayes (Post Tuning): Test AUC Curve - 46 -
Figure 80. Naive Bayes (Post Tuning): Train Confusion Matrix - 46 -
Figure 81. Naive Bayes (Post Tuning): Train Classification Report - 46 -
Figure 82. Naive Bayes (Post Tuning): Test Confusion Matrix - 46 -
Figure 83. Naive Bayes (Post Tuning): Test Classification Report - 47 -

Back to Top

Figure 84. KNN (Post Tuning): Train AUC Curve ..... - 48 -
Figure 85. KNN (Post Tuning): Test AUC Curve ..... - 48 -
Figure 86. KNN (Post Tuning): Train Confusion Matrix ..... - 48 -
Figure 87. KNN (Post Tuning): Train Classification Report ..... - 49 -
Figure 88. KNN (Post Tuning): Test Confusion Matrix ..... - 49 -
Figure 89. KNN (Post Tuning): Test Classification Report ..... - 49 -
Figure 90. Roosevelt Speech: Most Common Words ..... - 52 -
Figure 91. Kennedy Speech: Most Common Words ..... - 52 -
Figure 92. Nixon Speech: Most Common Words ..... - 52 -
Figure 93. Roosevelt Speech: WorldCloud ..... - 53 -
Figure 94. Kennedy Speech: WorldCloud ..... - 54 -
Figure 95. Nixon Speech: WorldCloud ..... - 55 -

Back to Top

Chapter 1. Problem 1

1.1 Problem Statement

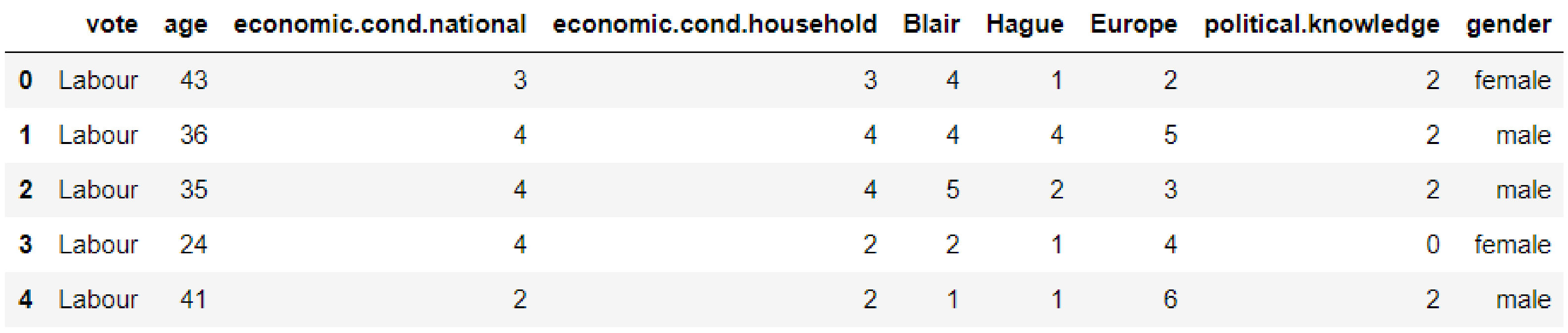
You are hired by one of the leading news channels CNBE who wants to analyze recent elections. This survey was conducted on 1525 voters with 9 variables. You have to build a model, to predict which party a voter will vote for on the basis of the given information, to create an exit poll that will help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

1.1.1 Data Dictionary:

Variable Name	Description
Vote	Party choice: Conservative or Labour
Age	Age of the voter (in years)
economic.cond.national	Assessment of current national economic conditions, 1 to 5
economic.cond.household	Assessment of current household economic conditions, 1 to 5
Blair	Assessment of the Labour leader, 1 to 5
Hague	Assessment of the Conservative leader, 1 to 5
Europe	An 11-point scale that measures respondents' attitudes toward European integration. High scores represent ‘Eurosceptic’ sentiment.
political.knowledge	Knowledge of parties' positions on European integration, 0 to 3.
gender	female or male

1.1.2 Read the dataset. Do the descriptive statistics and do the null value condition check. Write an inference on it.

Table 1 Dataframe: df (with head function)



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

Back to Top

Table 2 Dataframe: df (with describe function)

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

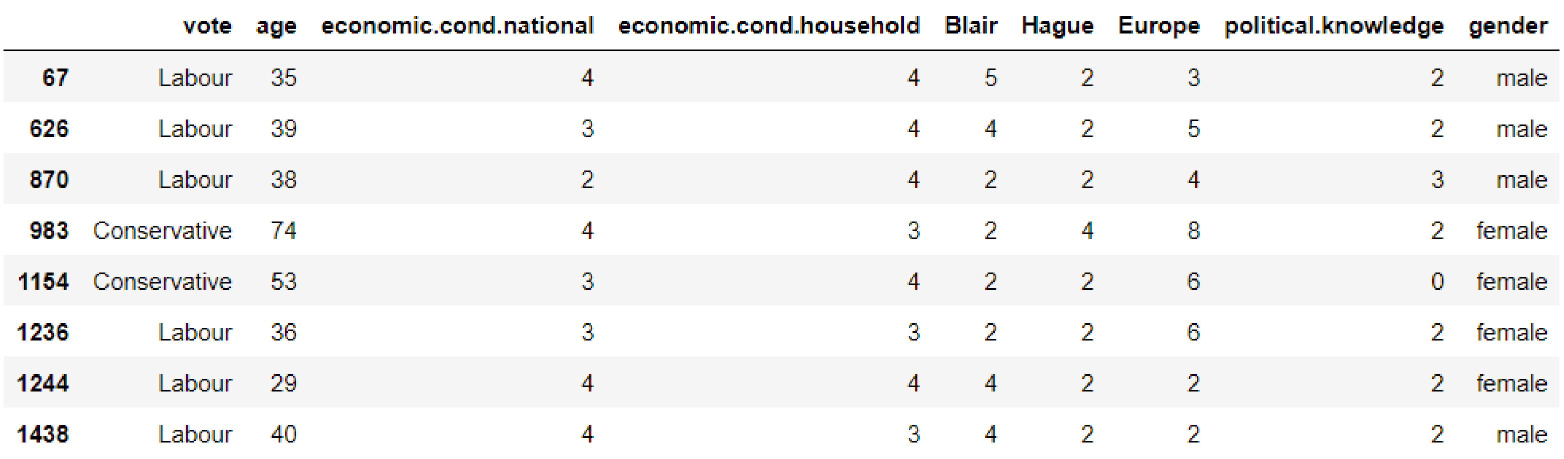
Figure 1. Dataset information

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

The dataset has a depth variable which has no null values. If we look at the data types we have object and integer types.

Back to Top

Table 3 Duplicate entries in df



	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

- **Categorical variables:**
  - The **'vote'** variable has 2 unique values, of which the (votes for the) "Labour" party dominates the observations
  - The **'Gender'** variable as expected has 2 unique values. The count of female voters is higher than male voters.
- **Numerical variables**:
  - **Age:** The mean age of the voters is approximately 54, which is almost equal to the median age (53). The age group of the sample size has range of 24 to 93.
  - **economic.cond.national:** It can be summarized that the general consensus of the sample size points towards a neutral opinion of the national economic condition (mean 3.2 and median 3). There are people who have a bad opinion of the national economic condition (score 1) as well as a very good opinion for the same (score 5).
  - **economic.cond.household:** It can be summarized that the general consensus of the sample size points towards a neutral opinion of the household economic condition (mean 3.1 and median 5). There are people who have a bad opinion of the household condition (score 1) as well as a very good opinion for the same (score 5).
  - **Blair vs. Hague:** We can conclude that the participants in the sample rate the Labour leader Blair (mean rating 3.3) higher than the Conservative leader Hague (mean rating 2.7)
  - **Europe:** The mean score (6.7) and median (6) hints that the overall attitude of the sample size leans slightly towards a 'Eurosceptic' sentiment, i.e., opposition of close ties between Britain and the European Union (EU)
  - **political.knowledge:** A median score of 2 indicates that a considerable amount of participants could have a fair knowledgeable about their respective parties' stand on European integration. However, since the 1st quartile also points at a score of 0, there could be a significant number of people who do not know about their parties' inclination.
  - **Eight duplicate rows** were detected. They have been removed from the dataset.

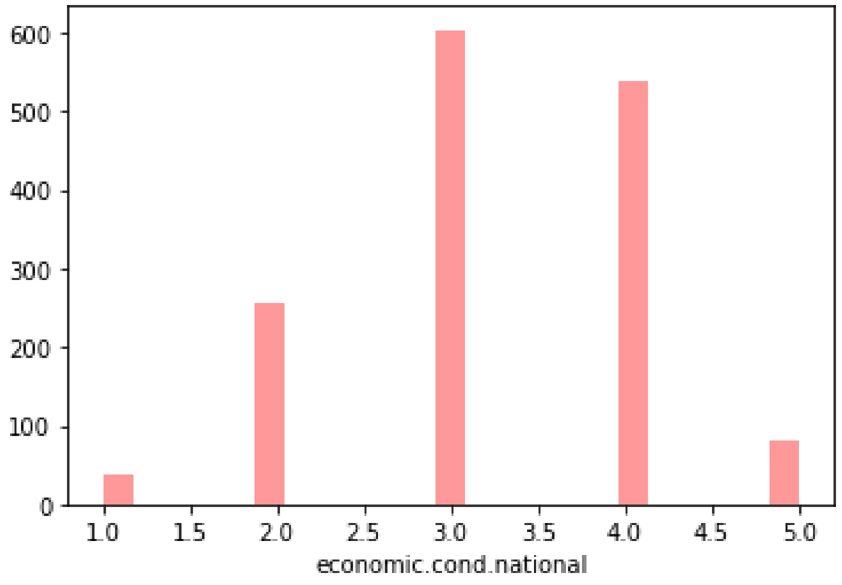
Back to Top

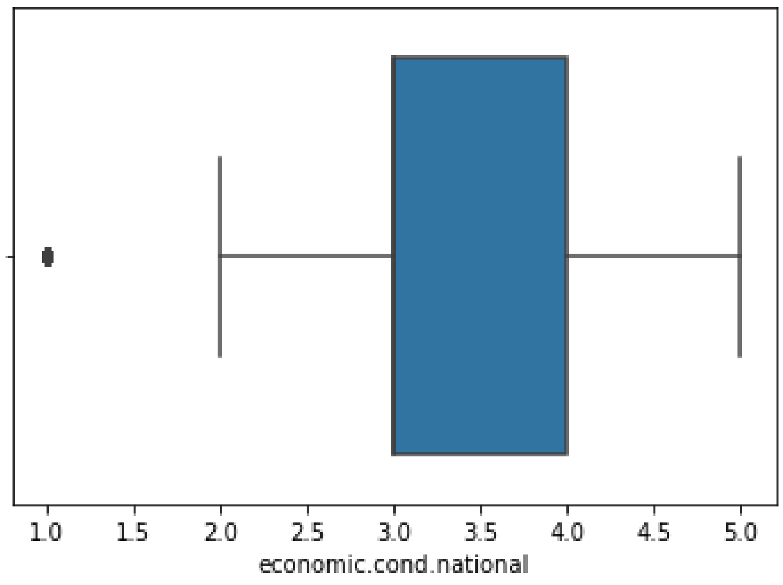
1.1.3 Perform Univariate and Bivariate Analysis. Do exploratory data analysis. Check for Outliers.

Figure 2. Skewness

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

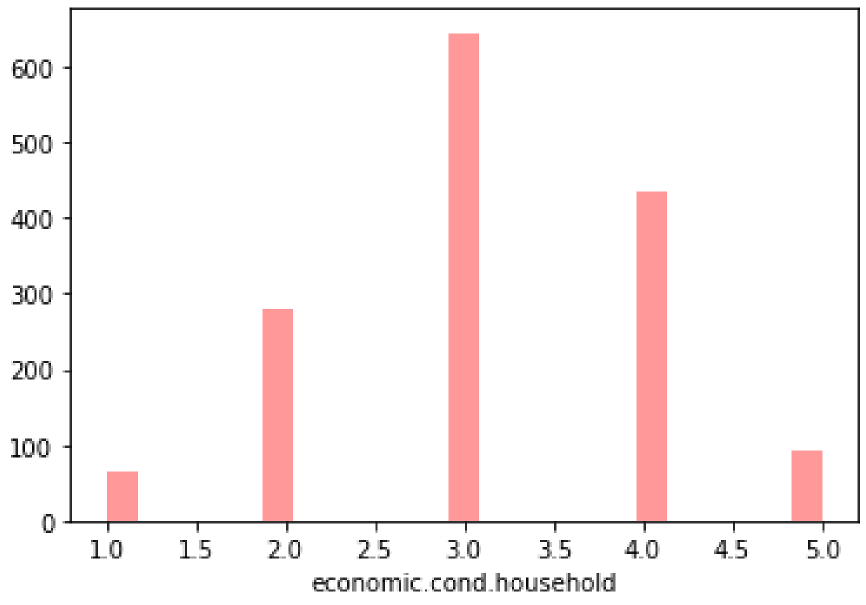
Figure 3. National economic condition data series: Description & graphical representation

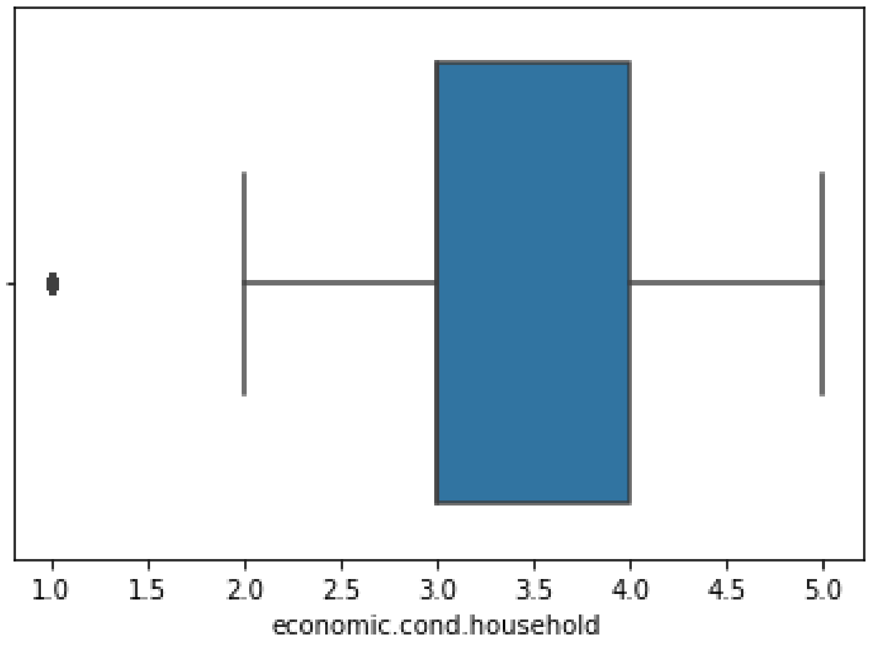




Back to Top

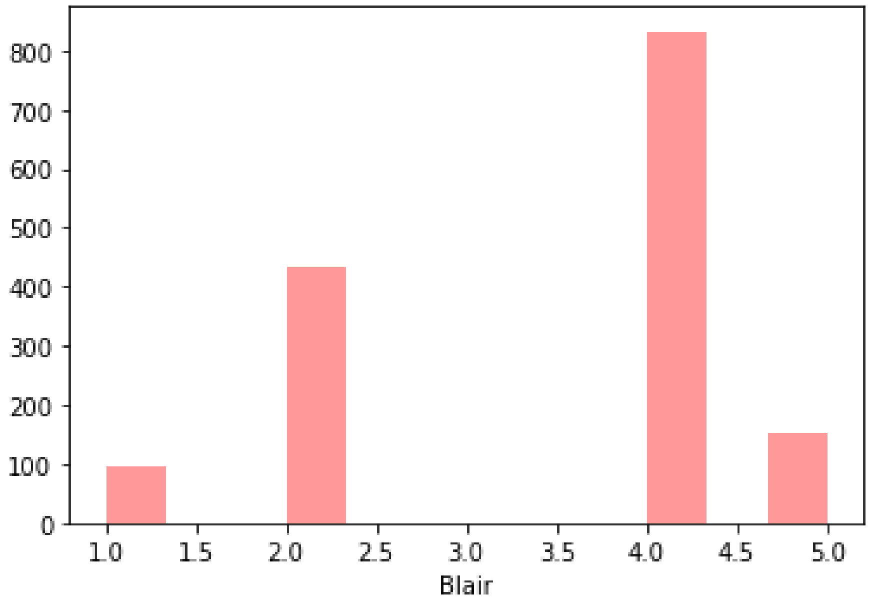
Figure 4. **Household economic condition data series: Description & graphical representation**





Back to Top

Figure 5. **Blair data series: Description & graphical representation**



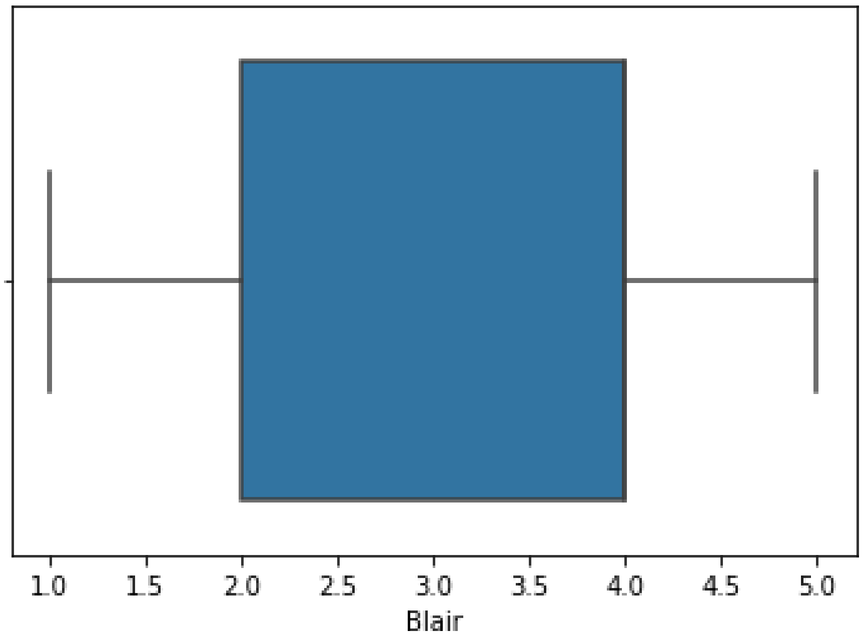
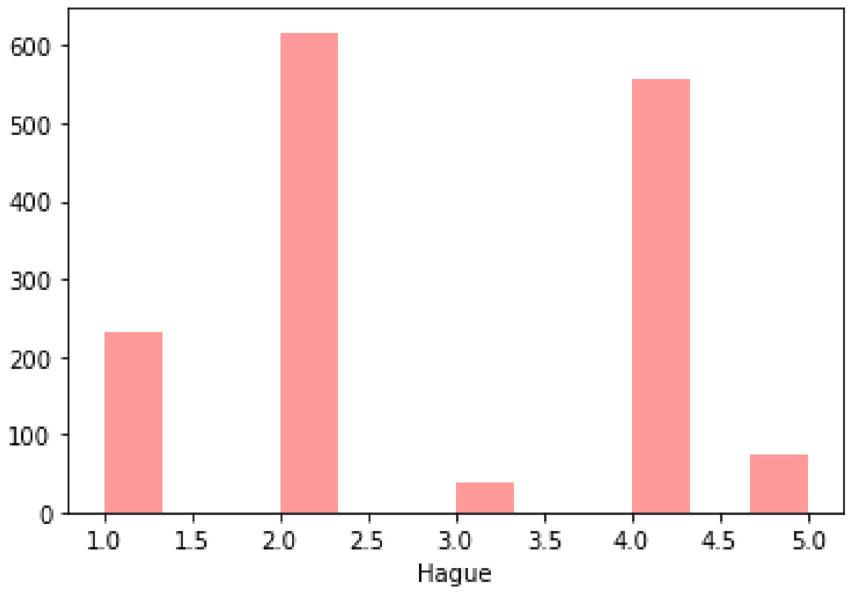
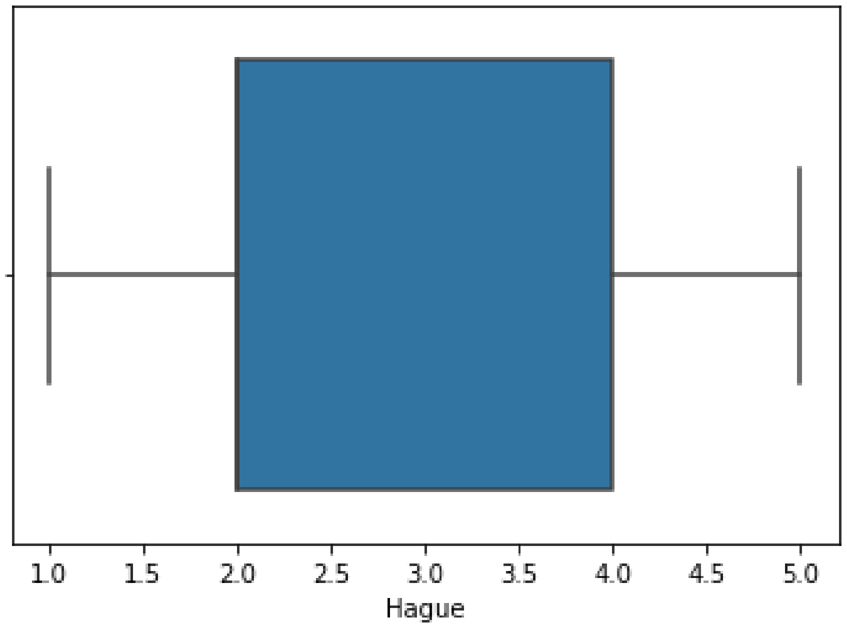


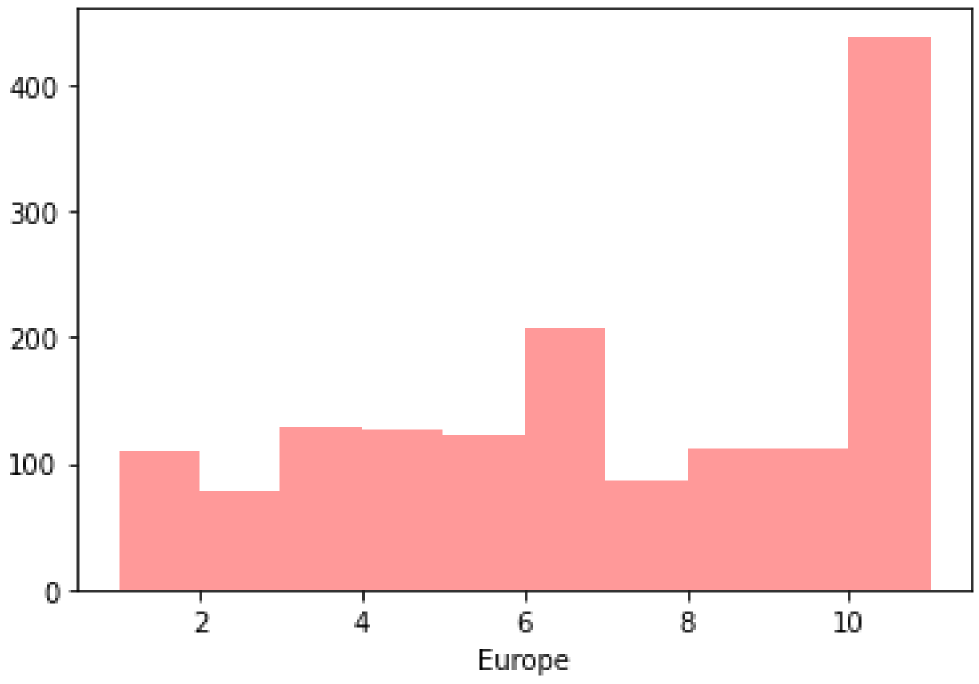
Figure 6. Hague data series: Description & graphical representation

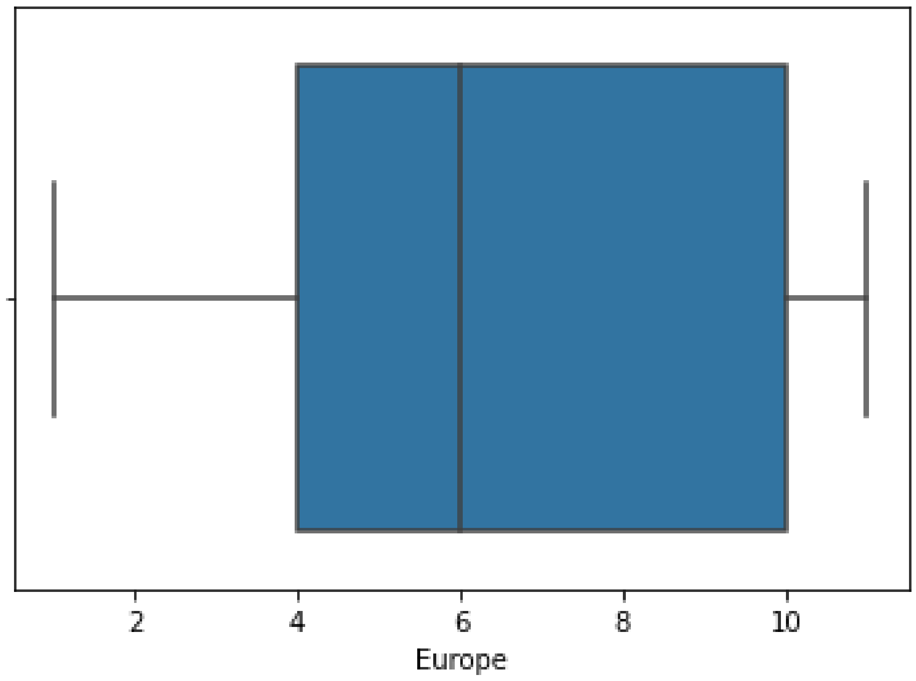




Back to Top

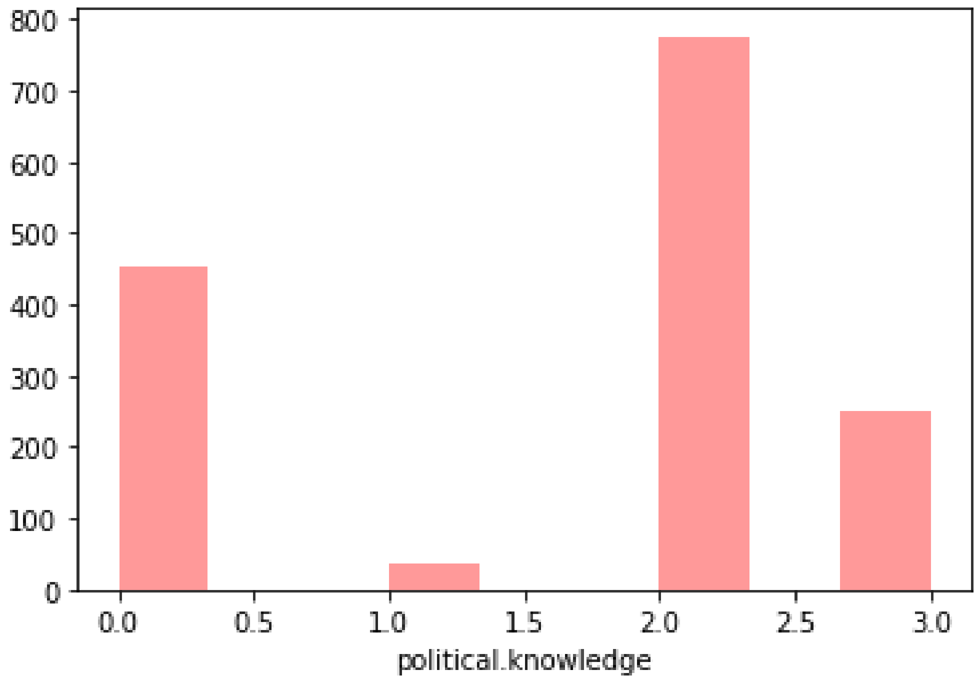
Figure 7. Europe data series: Description & graphical representation

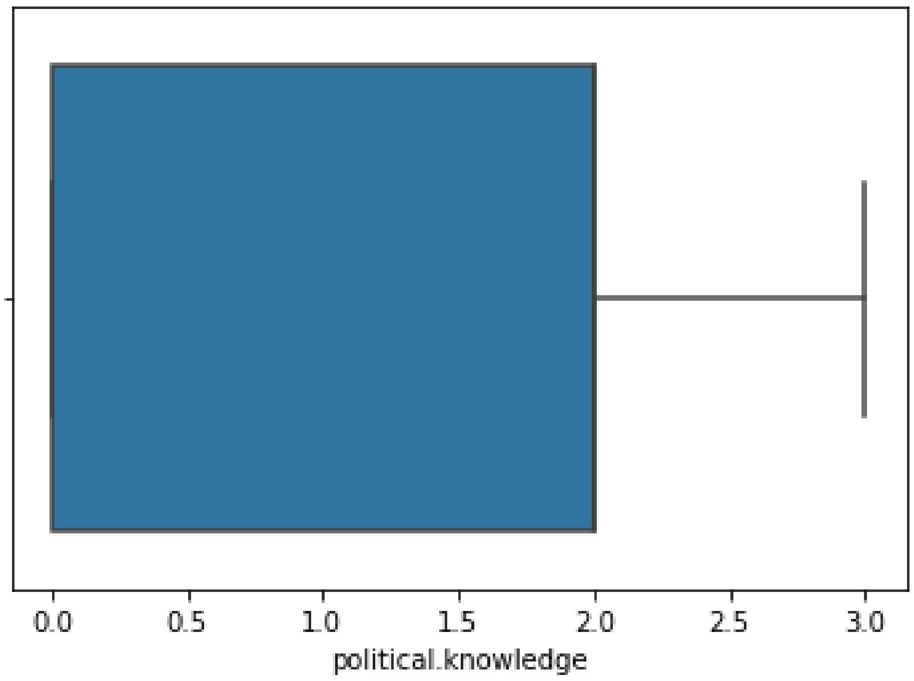




Back to Top

Figure 8. Political knowledge data series: Description & graphical representation





Back to Top

Figure 9. Pie chart analysis of the three variables (vote, votes for blair, votes for hague)

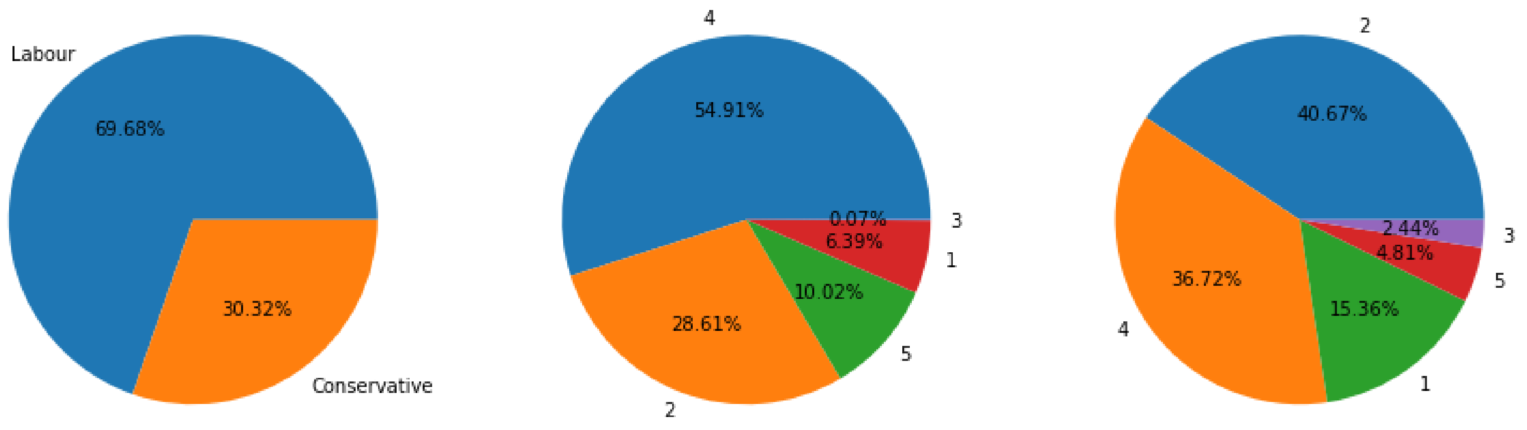
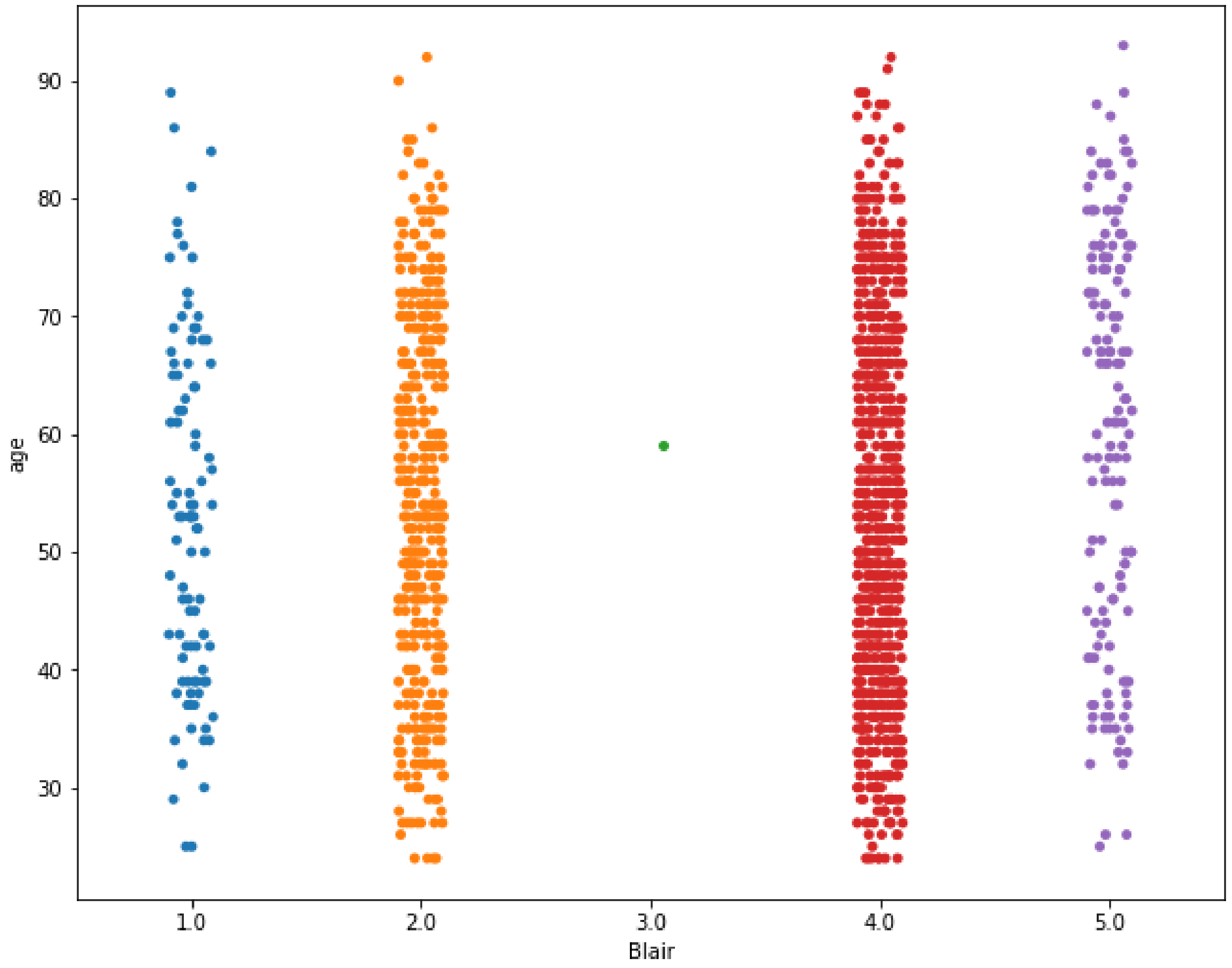
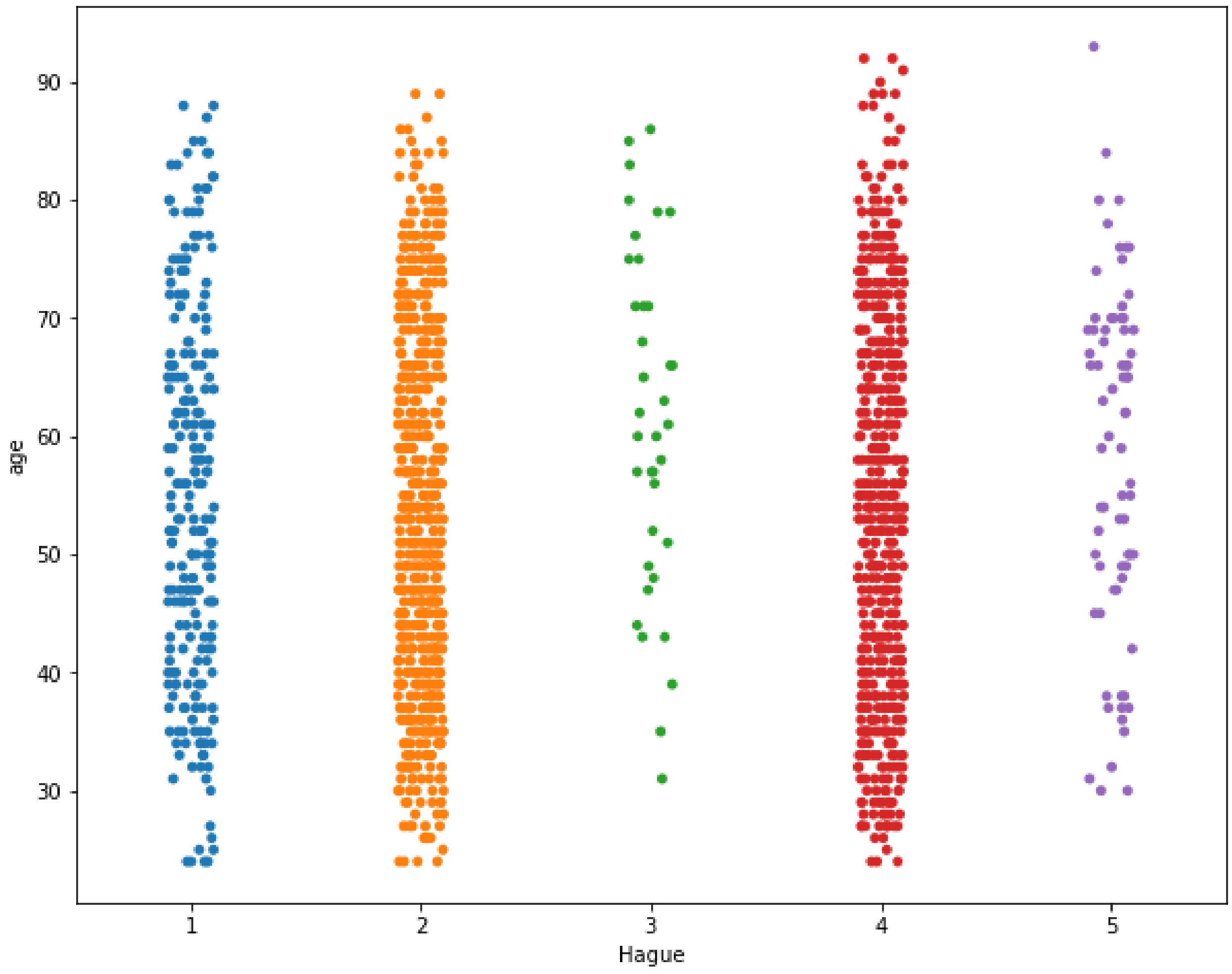


Figure 10. Strip chart analysis of blair and age



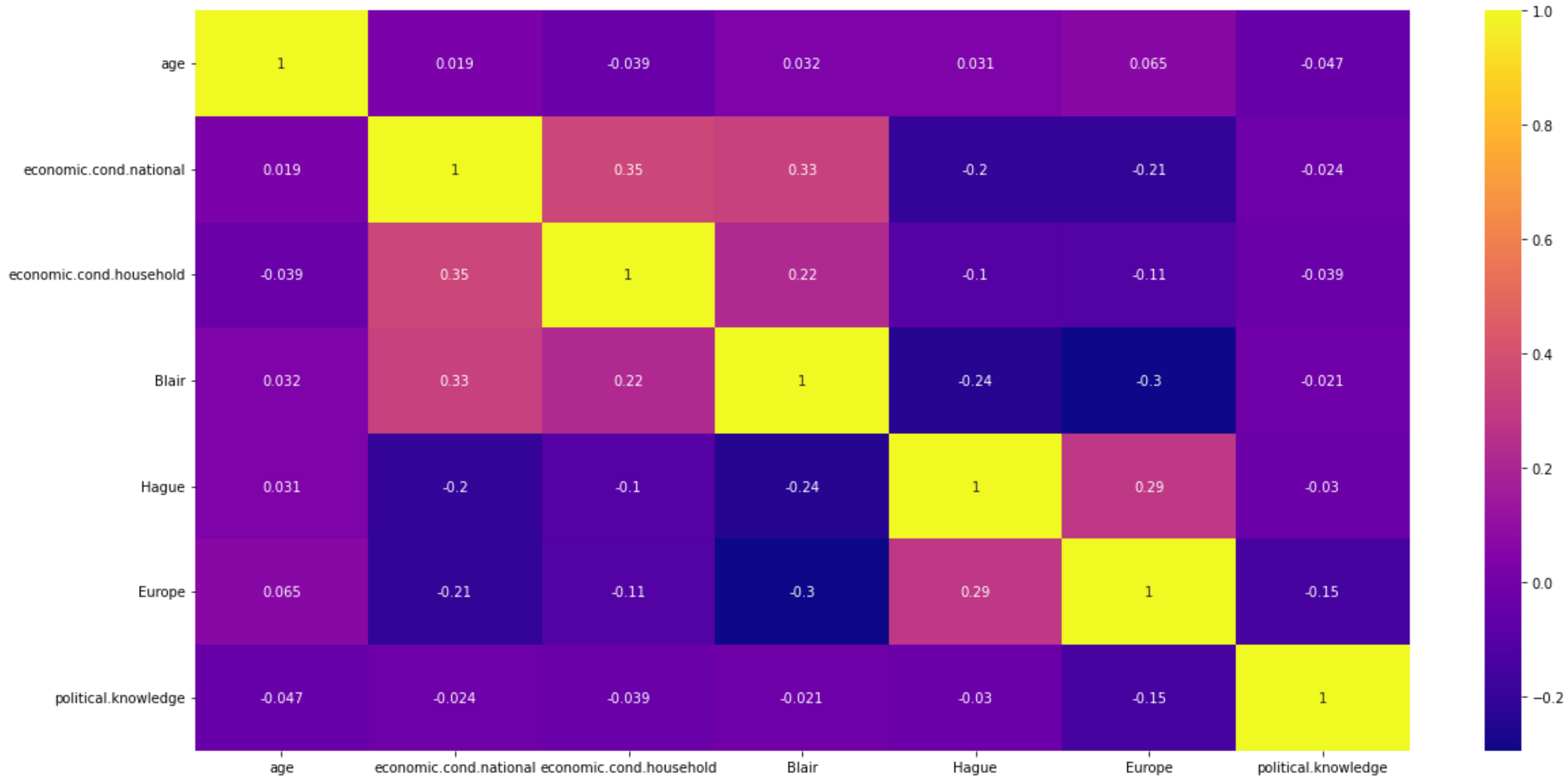
Back to Top

Figure 11. **Strip chart analysis of hague and age**



Back to Top

Figure 12. Correlation matrix



As seen in the above heatmap, there is negligible correlation between the variables.

Younger people seem to prefer the Labour party, whereas older people prefer the Conservative party.

People having an optimistic view of the national economic condition (scores of 3 and above) seem to prefer the Labour party; whereas the ones having a pessimistic view (scores of 3 and below) prefer voting for the Conservative party.

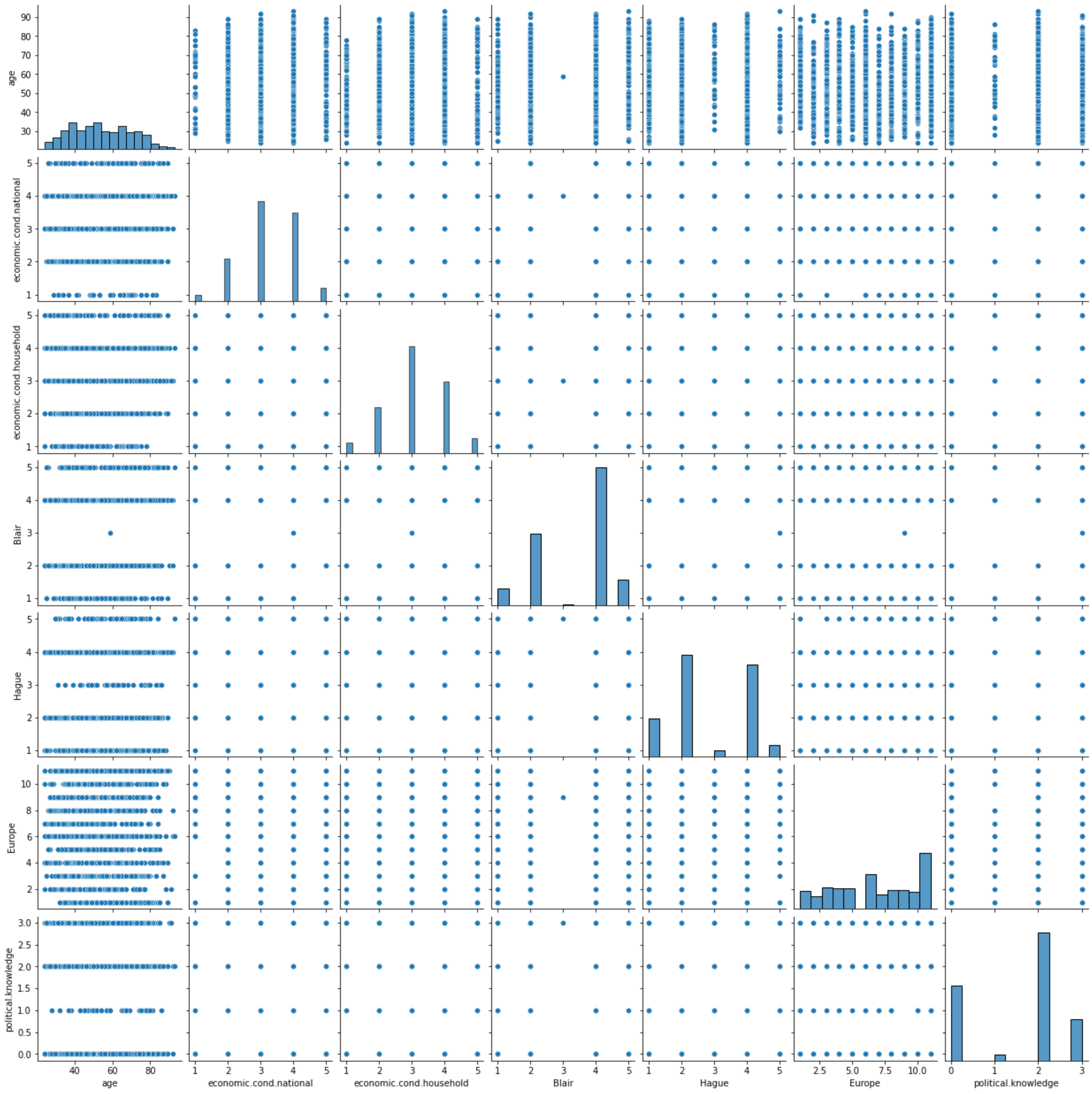
As evident, the people who have rated Blair highly (median rating of 4, plus a few outliers of 5) have voted for the Labour party (Blair being

the Labour party's leader). There are also a few people who have not rated Blair highly (ratings of 1 and 2), but have still gone on to vote for him.

People not having such a good assessment of Blair have voted for the Conservative party. Here, again there are a few exceptions (people who have rated Blair 4, but have gone on to vote for the Conservative party)

Back to Top

Figure 13. Pairplot



Back to Top

Figure 14. **Boxplot (with outliers)**

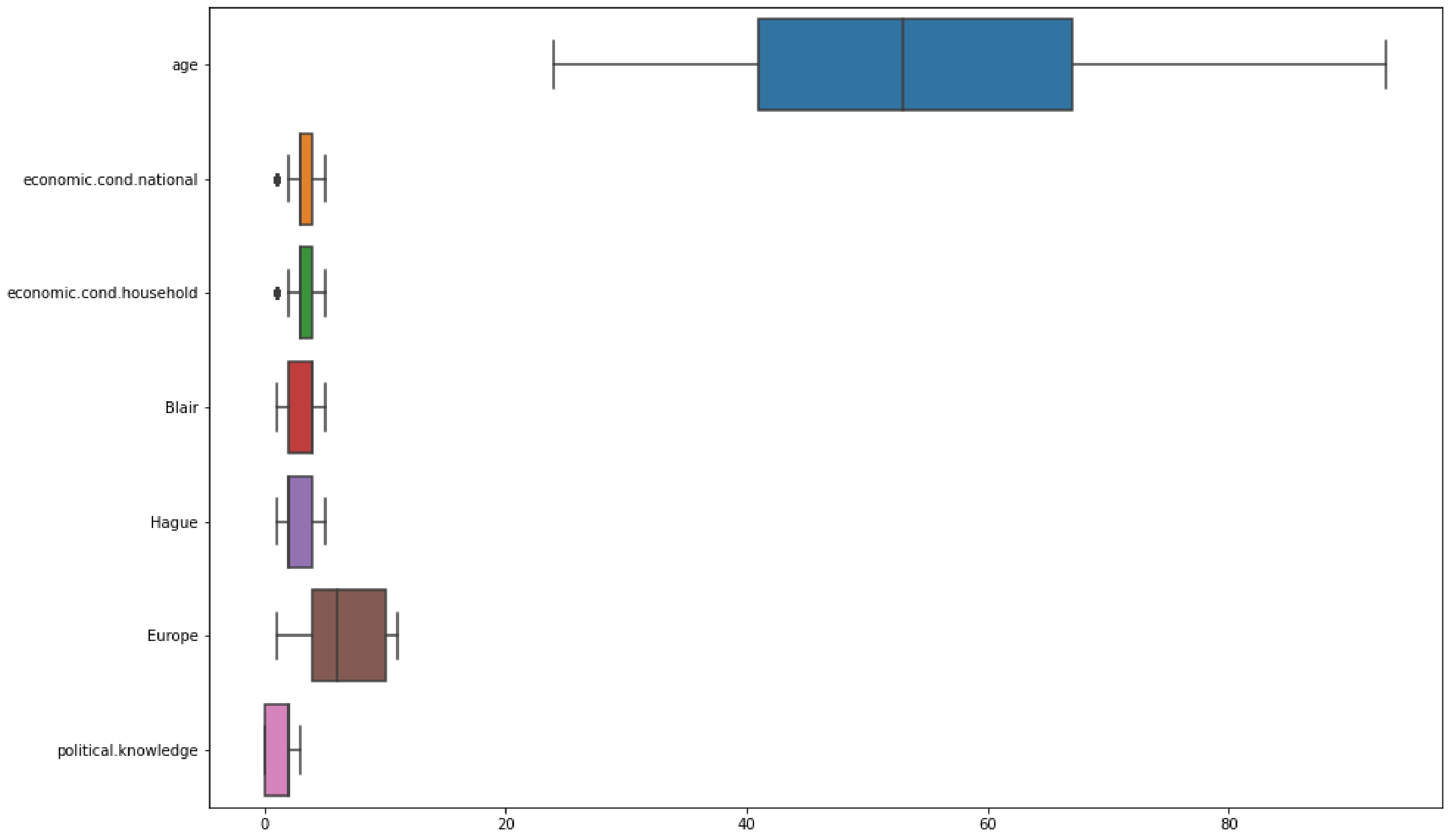
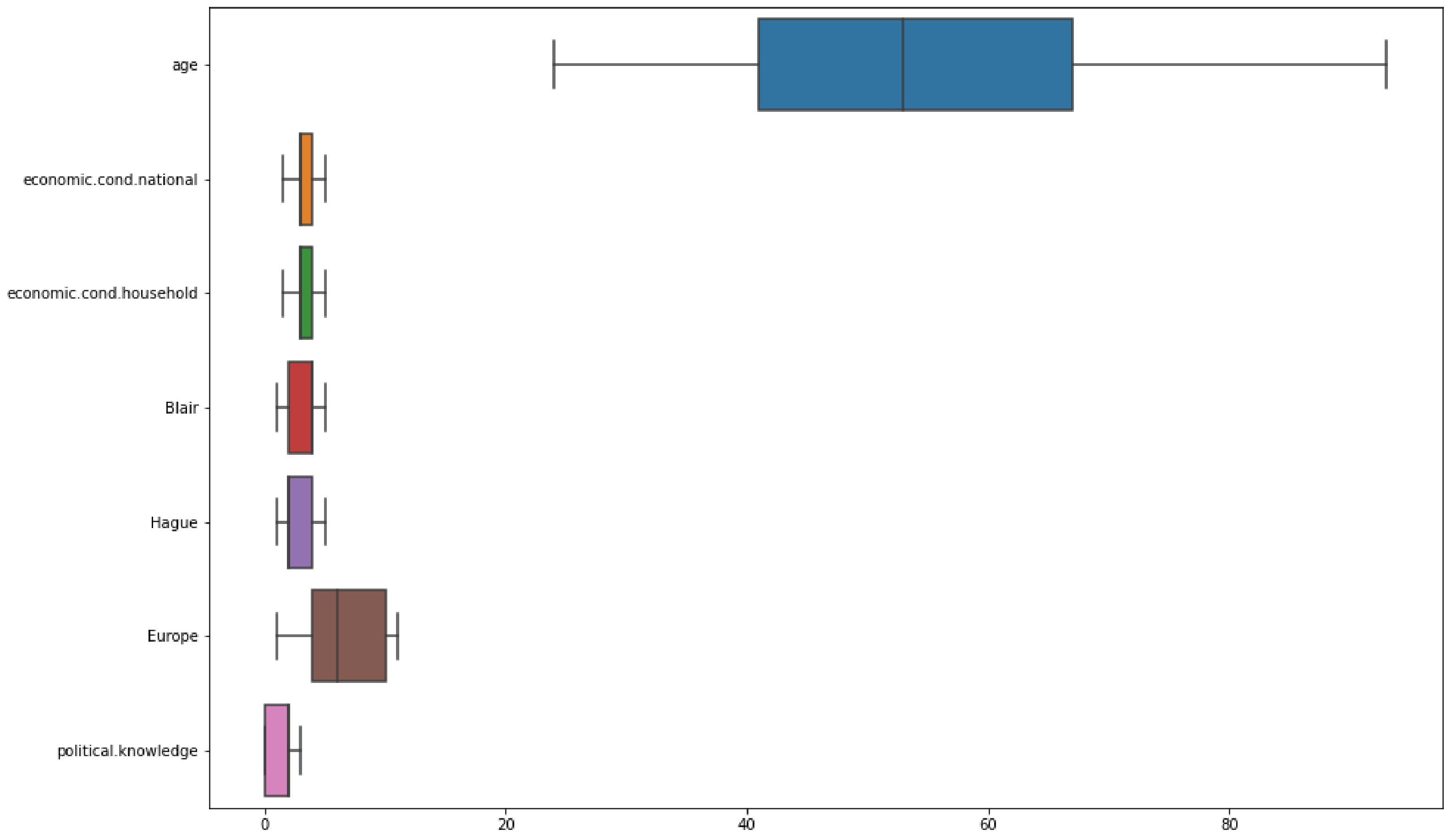


Figure 15. Boxplot (after outlier treatment)



1.1.4 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).

Figure 16. New Dataframe: df1 (with dummies)

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	vote_Labour	gender_male
0	43.0	3.0	3.0	4.0	1.0	2.0	2.0	1	0
1	36.0	4.0	4.0	4.0	4.0	5.0	2.0	1	1
2	35.0	4.0	4.0	5.0	2.0	3.0	2.0	1	1
3	24.0	4.0	2.0	2.0	1.0	4.0	0.0	1	0
4	41.0	2.0	2.0	1.0	1.0	6.0	2.0	1	1

- **Encoding categorical data:**
  - We use the one-hot encoding method by using the pd.get_dummies function. Also, we have opted for drop_first = True to prevent the creation of too many columns. Thus, two columns have been created:

Back to Top

  - vote_Labour: (“1” indicates that the respondent voted for the Labour party and “0” indicates that the respondent did not vote for the Labour party, i.e., voted for the Conservative party) gender_male: (“1” indicates male, “0” indicates female)

- **Scaling:**
  - We have discrete variables in the data set, with the respective scores ranging from 0 to 11 across all variables.
  - The other variable is “age”, with a range of 24 to 93.
  - These parameters cannot be exactly concluded to be having enormously different magnitudes. Thus, it is not really necessary to scale the variables.
- **Data split:**
  - As depicted in the Jupyter notebook, the dependent variable “vote_Labour” has been separated into y dataframe, while all the predictor variables have been copied into the X dataframe.
  - X and y have been split into the train and test sets (70:30 ratio)

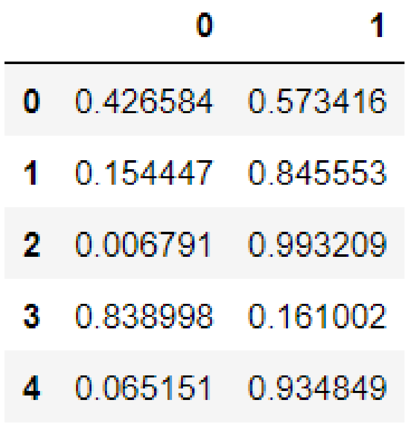
1.1.5 Apply Logistic Regression and LDA (linear discriminant analysis).

1.1.5.1 Logistic Regression

Logistic Regression: (Please refer to the Jupyter notebook)

- The LogisticRegression() function is fit on the X_train and y_train datasets
- The training and test datasets are then predicted
- The next step would be to get the predicted classes and probabilities:

Figure 17. Predicted Classes and Probabilities



	0	1
0	0.426584	0.573416
1	0.154447	0.845553
2	0.006791	0.993209
3	0.838998	0.161002
4	0.065151	0.934849

Back to Top

Figure 18. Logistic Regression: Train AUC Curve

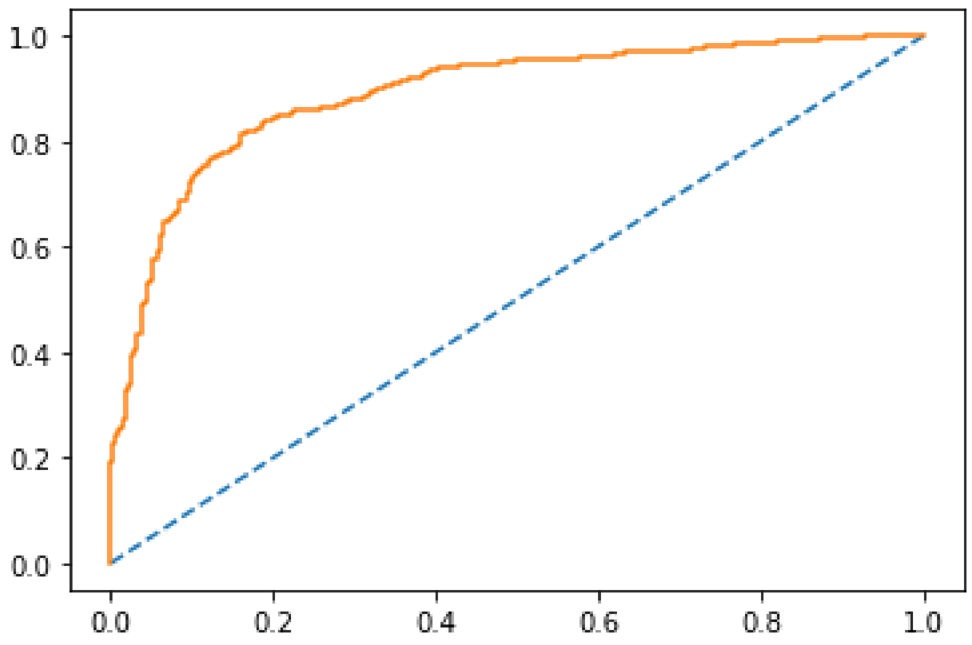


Figure 19. Logistic Regression: Test AUC Curve

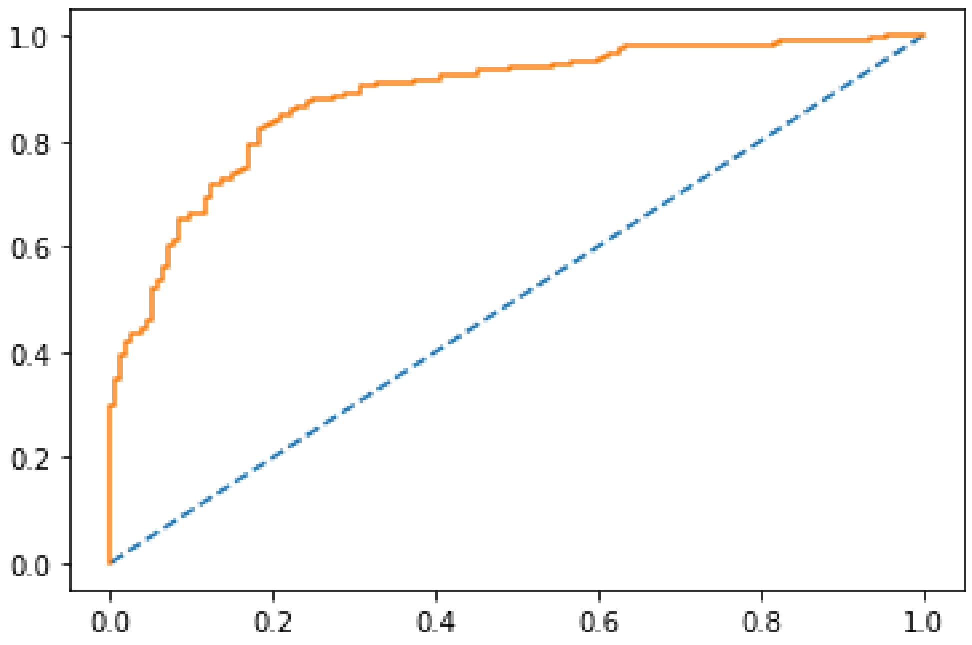
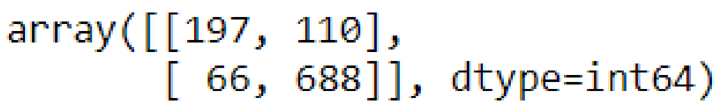
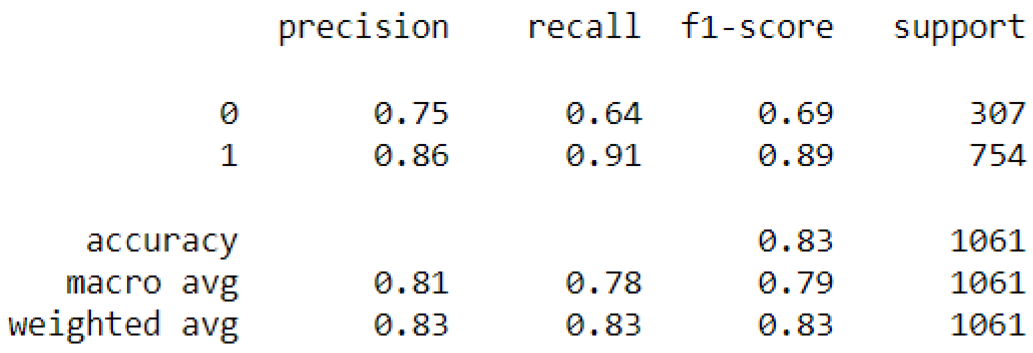


Figure 20. Logistic Regression: Train Confusion Matrix



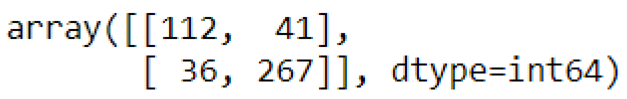
```
array([[197, 110],
       [ 66, 688]], dtype=int64)
```

Figure 21. Logistic Regression: Train Classification Report



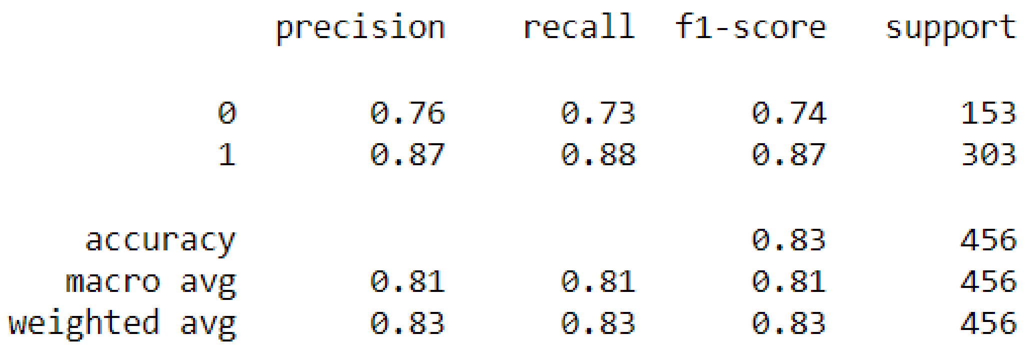
```
              precision    recall  f1-score   support

           0       0.75      0.64      0.69       307
           1       0.86      0.91      0.89       754

    accuracy                           0.83      1061
   macro avg       0.81      0.78      0.79      1061
weighted avg       0.83      0.83      0.83      1061
```

Figure 22. Logistic Regression: Test Confusion Matrix



```
array([[112,  41],
       [ 36, 267]], dtype=int64)
```

Figure 23. Logistic Regression: Test Classification Report



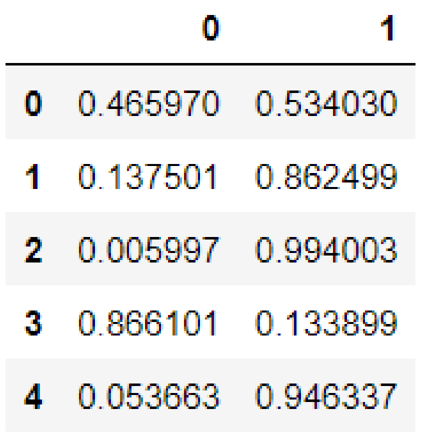
```
              precision    recall  f1-score   support

           0       0.76      0.73      0.74       153
           1       0.87      0.88      0.87       303

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

1.1.5.2 Linear Discriminant Analysis (LDA)

LDA: (Please refer to the Jupyter notebook)

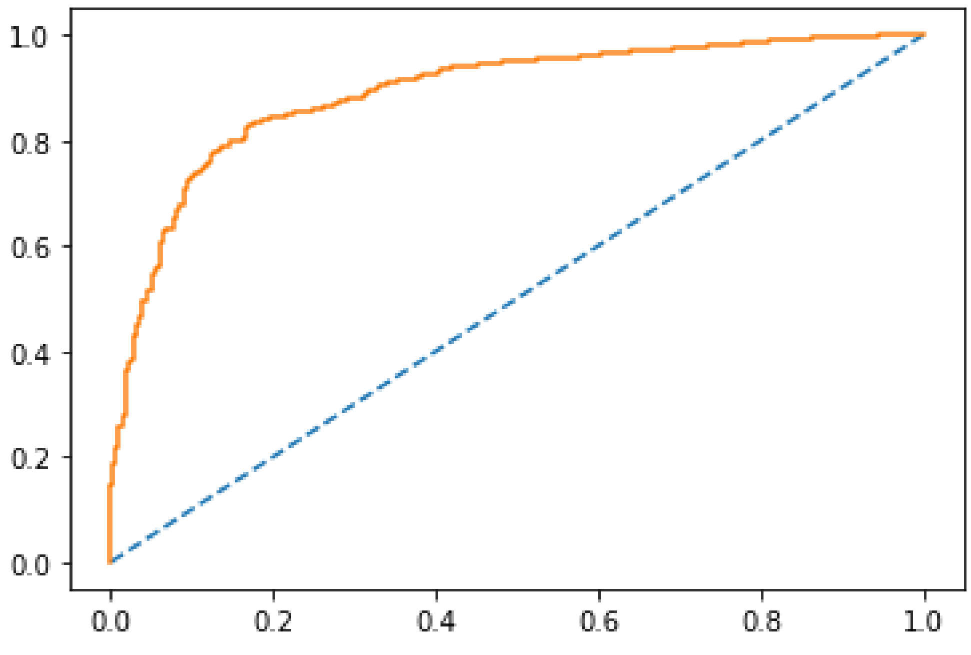
- The LinearDiscriminantAnalysis() function is fit on the X_train and y_train datasets
- The training and test datasets are then predicted
- The next step would be to get the predicted classes and probabilities:

Figure 24. Predicted Classes and Probabilities



	0	1
0	0.465970	0.534030
1	0.137501	0.862499
2	0.005997	0.994003
3	0.866101	0.133899
4	0.053663	0.946337

Figure 25. Linear Discriminant Analysis (LDA): Train AUC Curve



Back to Top

Figure 26. **Linear Discriminant Analysis (LDA): Test AUC Curve**

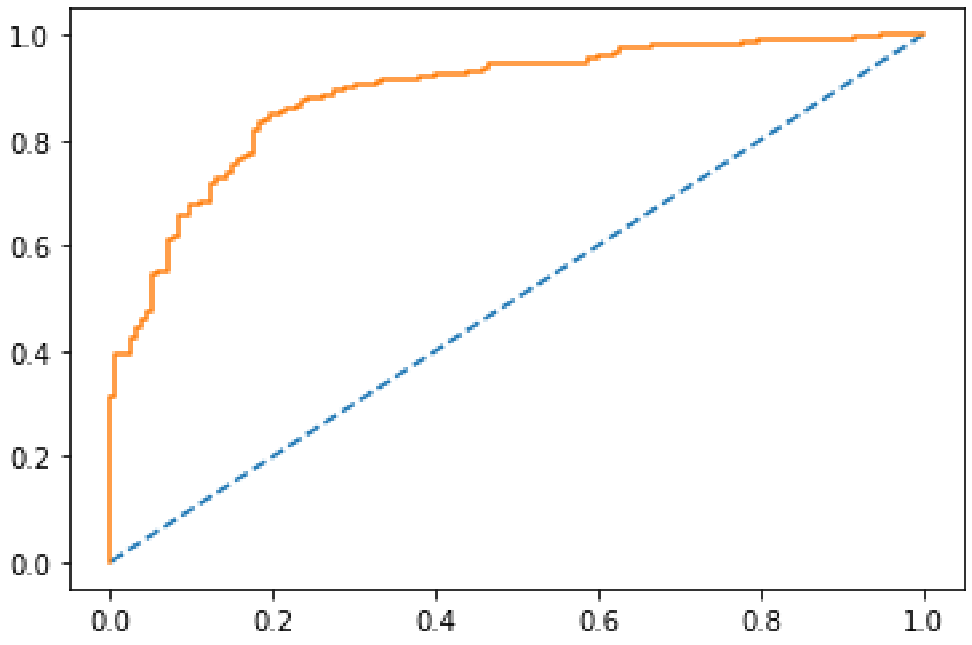
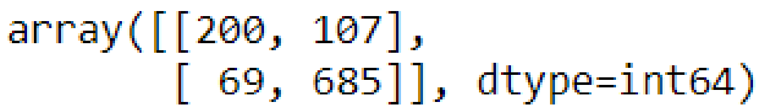
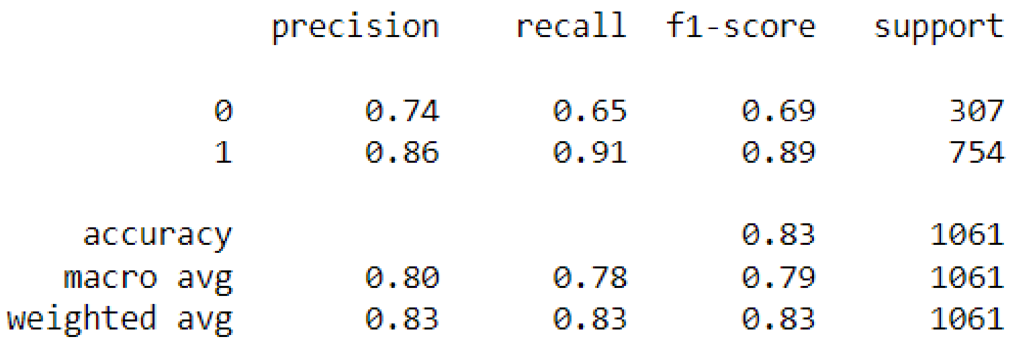


Figure 27. **Linear Discriminant Analysis (LDA): Train Confusion Matrix**



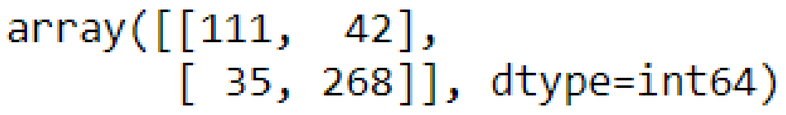
```
array([[200, 107],
       [ 69, 685]], dtype=int64)
```

Figure 28. **Linear Discriminant Analysis (LDA): Train Classification Report**



```
              precision    recall  f1-score   support

           0       0.74      0.65      0.69       307
           1       0.86      0.91      0.89       754

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

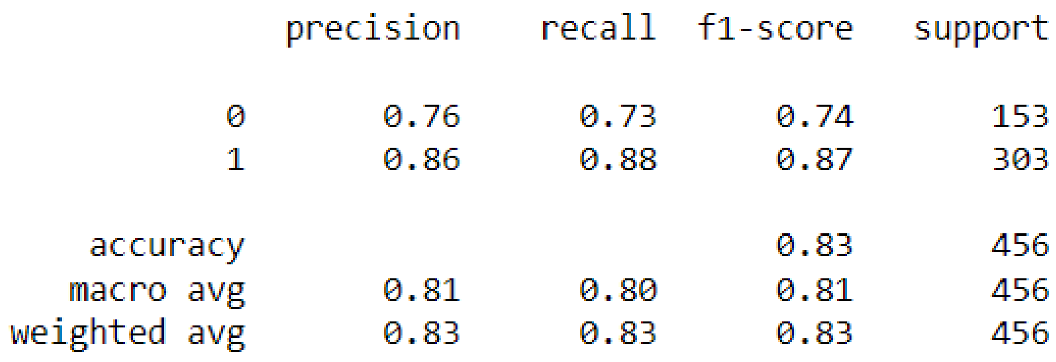
Figure 29. **Linear Discriminant Analysis (LDA): Test Confusion Matrix**



```
array([[111,  42],
       [ 35, 268]], dtype=int64)
```

Back to Top

Figure 30. Linear Discriminant Analysis (LDA): Test Classification Report



```
              precision    recall  f1-score   support

           0       0.76      0.73      0.74       153
           1       0.86      0.88      0.87       303

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

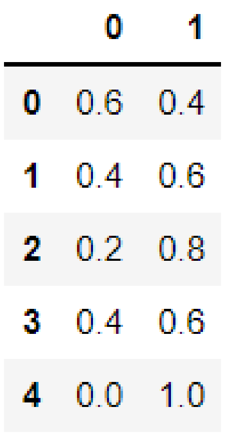
1.1.6 Apply KNN Model and Naïve Bayes Model. Interpret the results.

1.1.6.1 KNN

KNN: (Please refer to the Jupyter notebook)

- The KNeighborsClassifier () function is fit on the X_train and y_train datasets
- The training and test datasets are then predicted
- The next step would be to get the predicted classes and probabilities:

Figure 31. Predicted Classes and Probabilities



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

Figure 32. KNN: Train AUC Curve

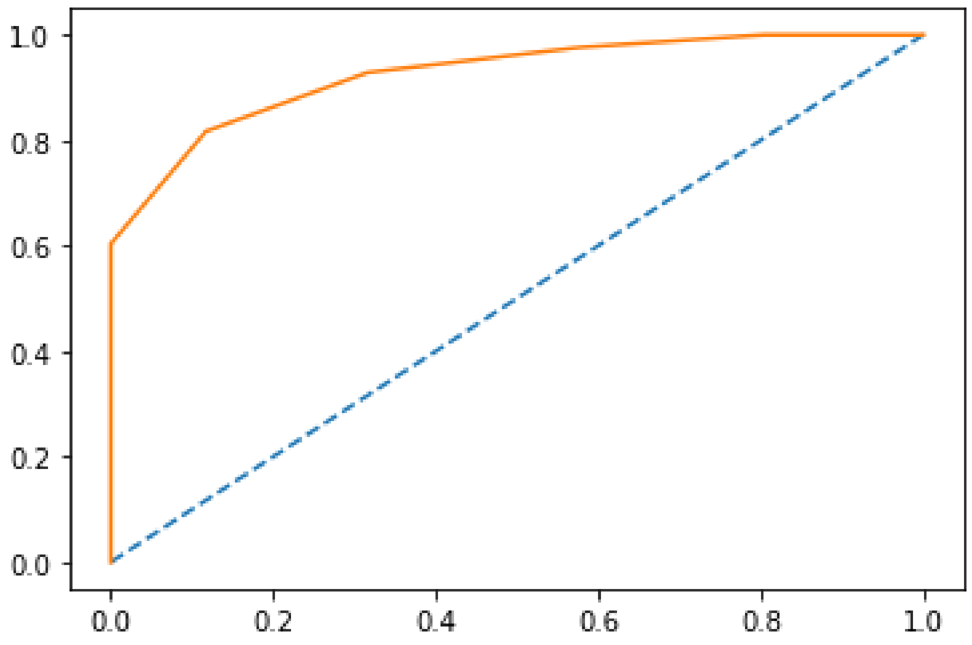


Figure 33. KNN: Test AUC Curve

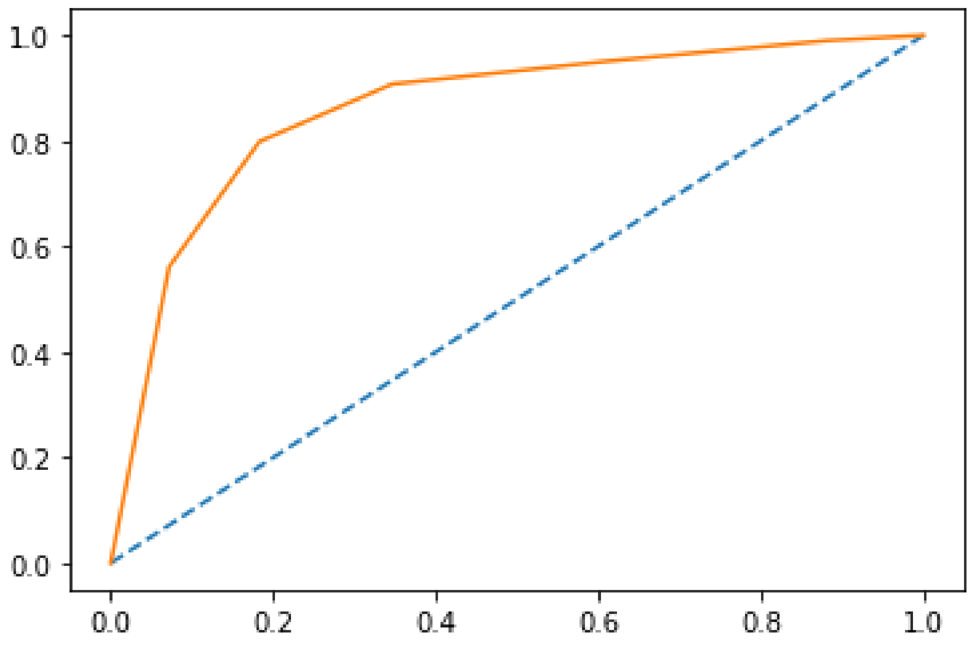


Figure 34. **KNN: Train Confusion Matrix**

```
array([[210,  97],
       [ 54, 700]], dtype=int64)
```

Figure 35. **KNN: Train Classification Report**

```
              precision    recall  f1-score   support

           0       0.80      0.68      0.74       307
           1       0.88      0.93      0.90       754

    accuracy                           0.86      1061
   macro avg       0.84      0.81      0.82      1061
weighted avg       0.85      0.86      0.85      1061
```

Figure 36. **KNN: Test Confusion Matrix**

```
array([[100,  53],
       [ 28, 275]], dtype=int64)
```

Figure 37. **KNN: Test Classification Report**

```
              precision    recall  f1-score   support

           0       0.78      0.65      0.71       153
           1       0.84      0.91      0.87       303

    accuracy                           0.82       456
   macro avg       0.81      0.78      0.79       456
weighted avg       0.82      0.82      0.82       456
```

Back to Top

1.1.6.2 Naïve Bayes

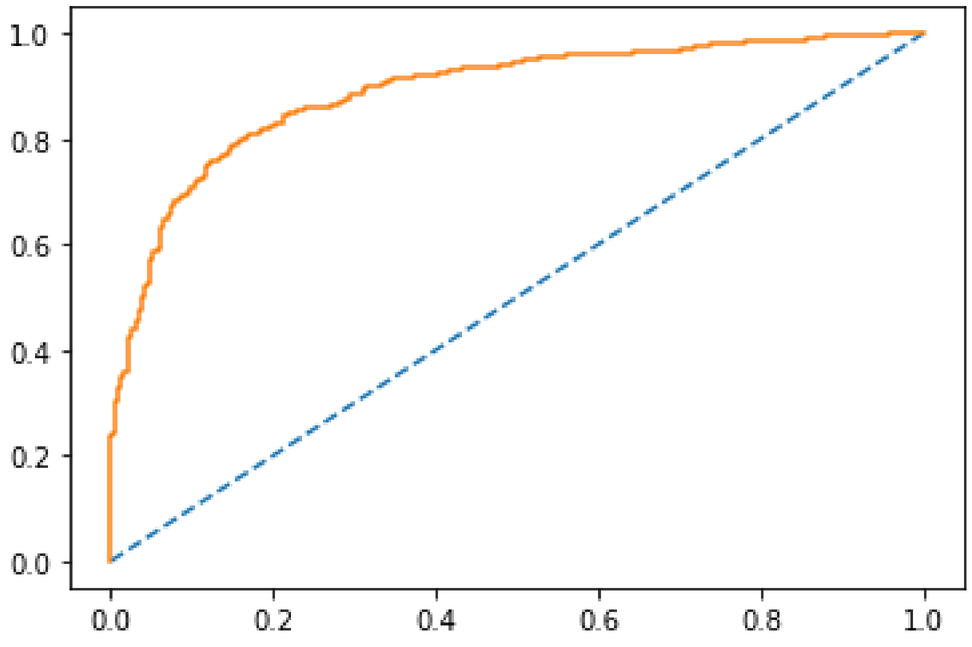
Naïve Bayes: (Please refer to the Jupyter notebook)

- The GaussianNB() function is fit on the X_train and y_train datasets
- The training and test datasets are then predicted
- The next step would be to get the predicted classes and probabilities:

Figure 38. Predicted Classes and Probabilities

	0	1
0	0.553915	0.446085
1	0.131740	0.868260
2	0.000247	0.999753
3	0.947290	0.052710
4	0.030269	0.969731

Figure 39. NB: Train AUC Curve



Back to Top

Figure 40. NB: Test AUC Curve

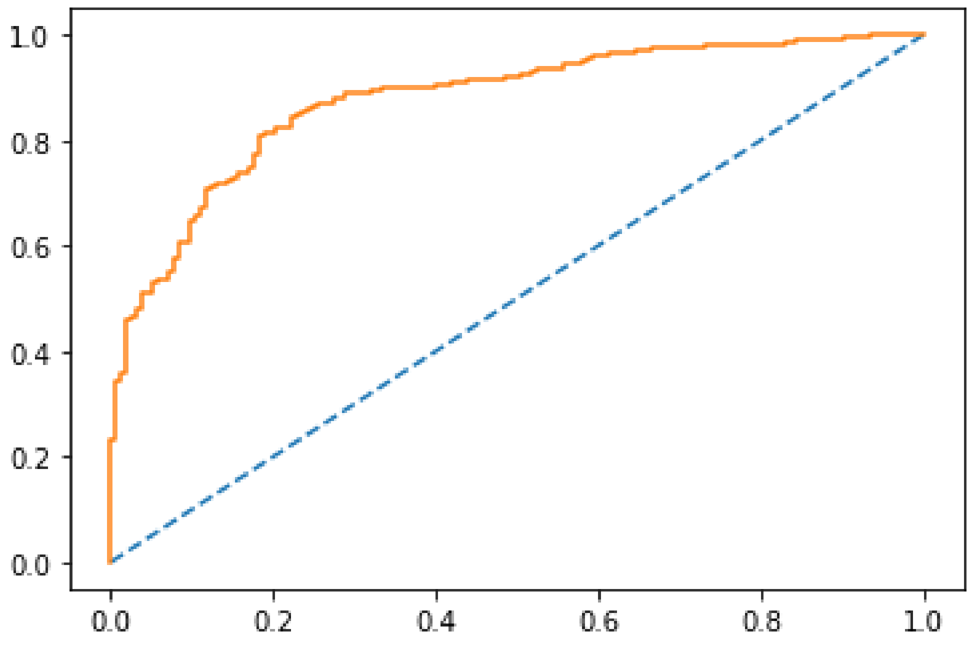
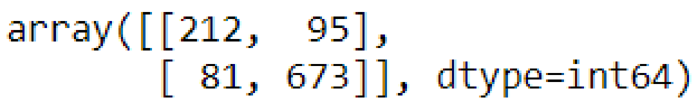
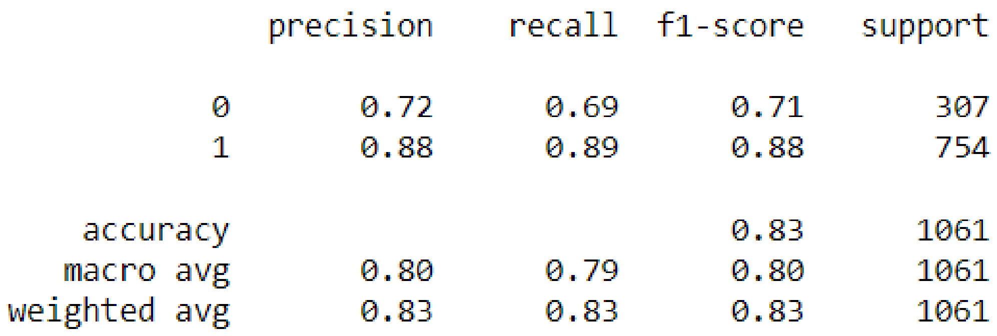


Figure 41. NB: Train Confusion Matrix



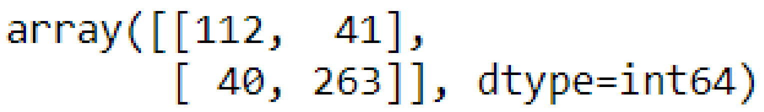
```
array([[212,  95],
       [ 81, 673]], dtype=int64)
```

Figure 42. NB: Train Classification Report



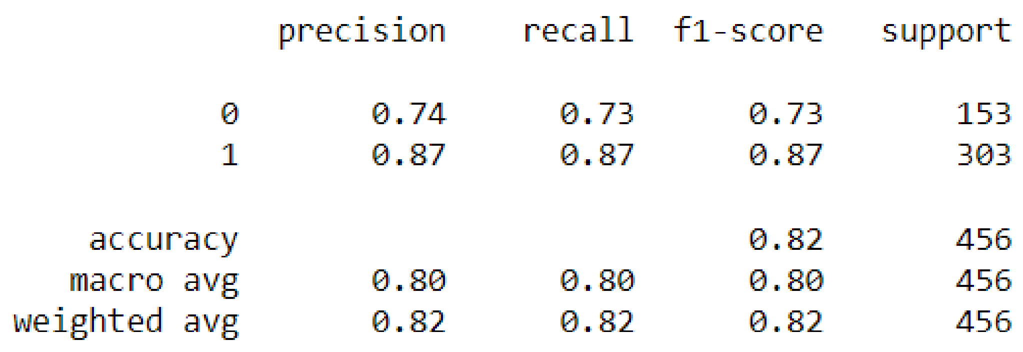
```
              precision    recall  f1-score   support

           0       0.72      0.69      0.71       307
           1       0.88      0.89      0.88       754

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

Figure 43. NB: Test Confusion Matrix



```
array([[112,  41],
       [ 40, 263]], dtype=int64)
```

Figure 44. NB: Test Classification Report



```
              precision    recall  f1-score   support

           0       0.74      0.73      0.73       153
           1       0.87      0.87      0.87       303

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

1.1.7 Model Tuning, Bagging (Random Forest should be applied for Bagging), and Boosting.

1.1.8 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.

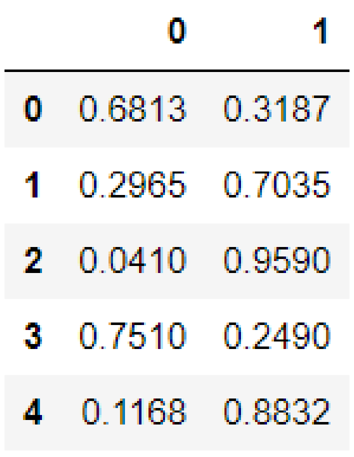
(Note: Combining two questions together)

1.1.8.1 Bagging (on Random Forest Classifier)

Bagging: (Please refer to the Jupyter notebook)

- The RandomForestClassifier() function is fit on the X_train and y_train datasets
- The training and test datasets are then predicted
- The next step would be to get the predicted classes and probabilities:

Figure 45. Predicted Classes and Probabilities



	0	1
0	0.6813	0.3187
1	0.2965	0.7035
2	0.0410	0.9590
3	0.7510	0.2490
4	0.1168	0.8832

Back to Top

Figure 46. Bagging: Train AUC Curve

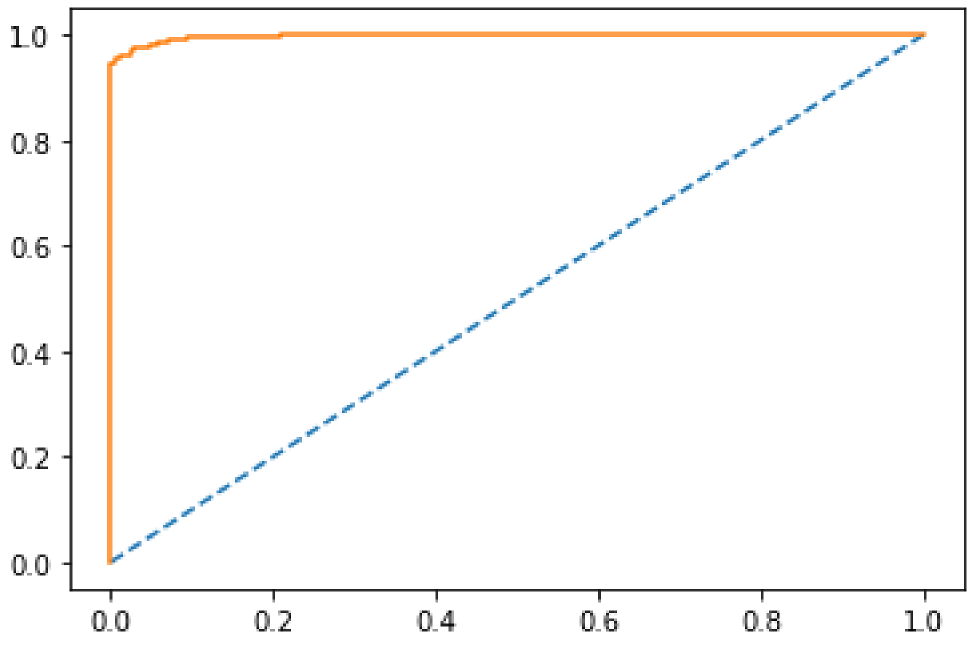


Figure 47. Bagging: Test AUC Curve

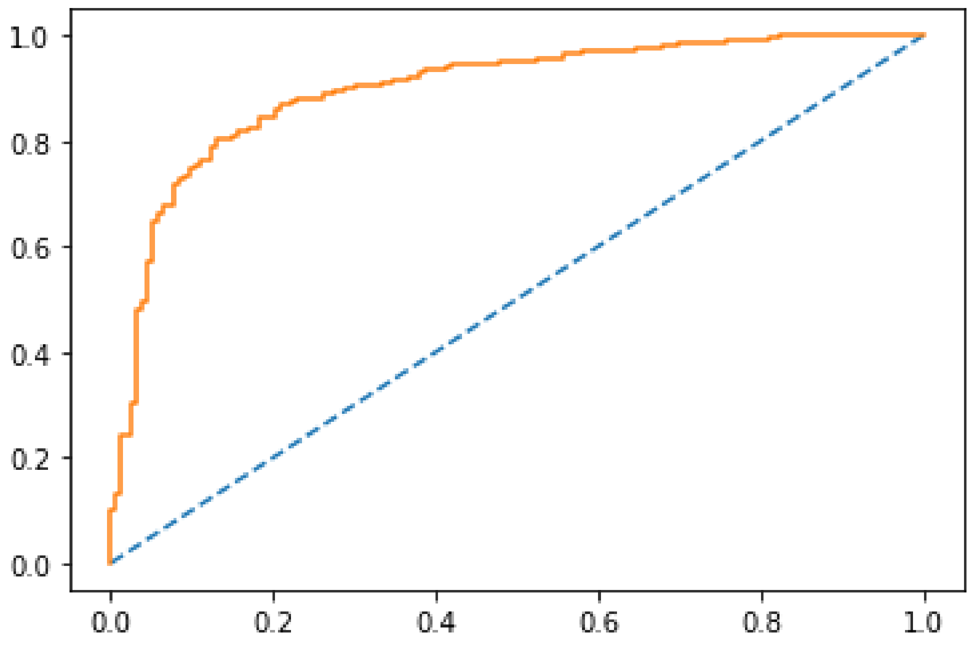
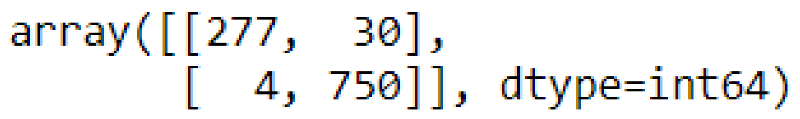
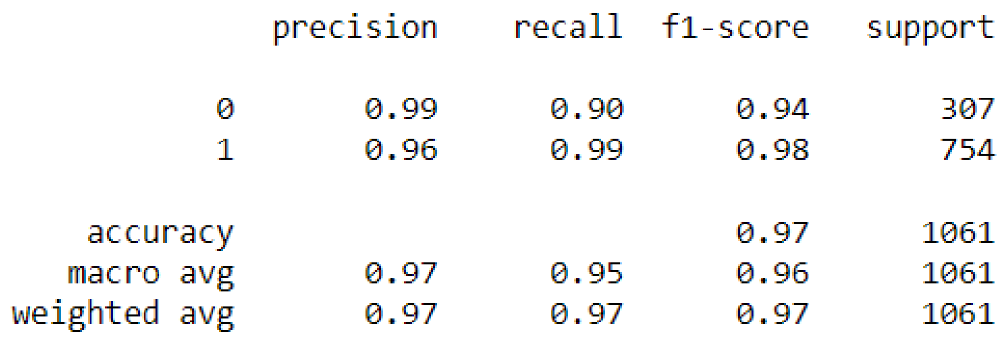


Figure 48. Bagging: Train Confusion Matrix



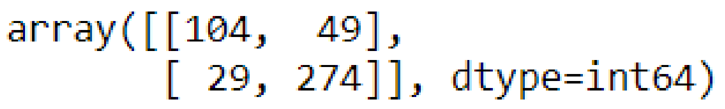
```
array([[277,  30],
       [  4, 750]], dtype=int64)
```

Figure 49. Bagging: Train Classification Report



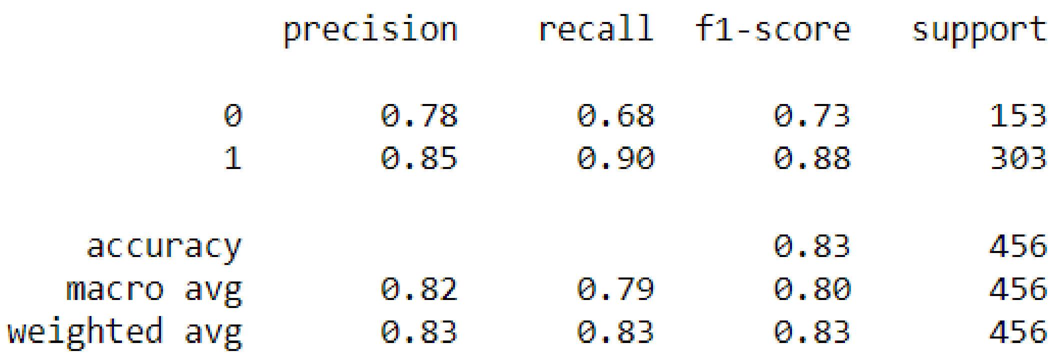
```
              precision    recall  f1-score   support

           0       0.99      0.90      0.94       307
           1       0.96      0.99      0.98       754

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

Figure 50. Bagging: Test Confusion Matrix



```
array([[104,  49],
       [ 29, 274]], dtype=int64)
```

Figure 51. Bagging: Test Classification Report



```
              precision    recall  f1-score   support

           0       0.78      0.68      0.73       153
           1       0.85      0.90      0.88       303

    accuracy                           0.83       456
   macro avg       0.82      0.79      0.80       456
weighted avg       0.83      0.83      0.83       456
```

1.1.8.2 ADA Boosting

ADA Boosting: (Please refer to the Jupyter notebook)

- The AdaBoostClassifier() function is fit on the X_train and y_train datasets
- The training and test datasets are then predicted
- The next step would be to get the predicted classes and probabilities:

Back to Top

Figure 52. Predicted Classes and Probabilities

	0	1
0	0.326483	0.673517
1	0.211437	0.788563
2	0.169235	0.830765
3	0.548865	0.451135
4	0.402257	0.597743

Figure 53. ADA Boosting: Train AUC Curve

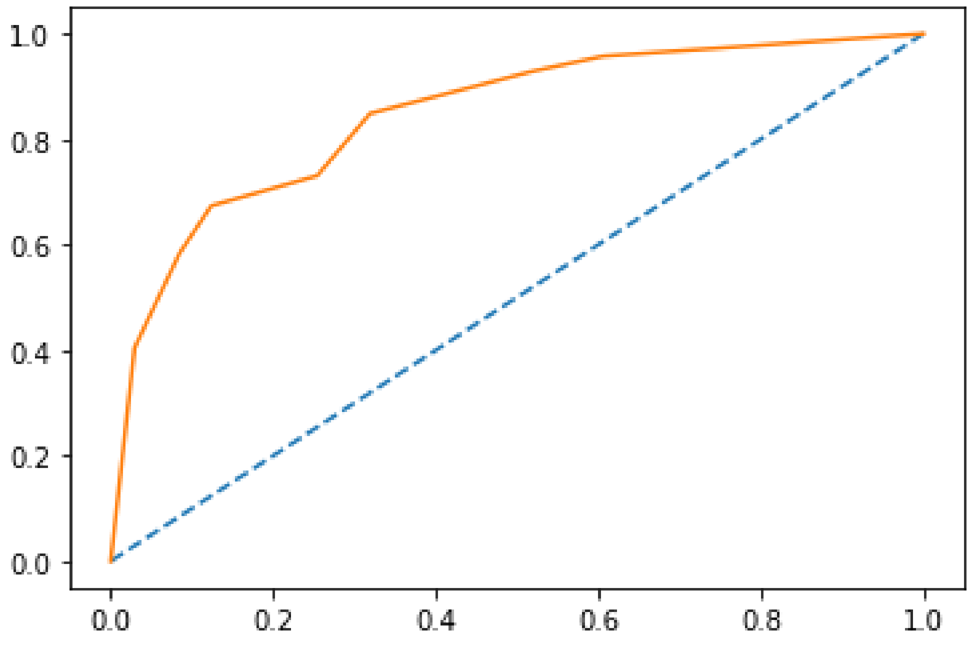


Figure 54. ADA Boosting: Test AUC Curve

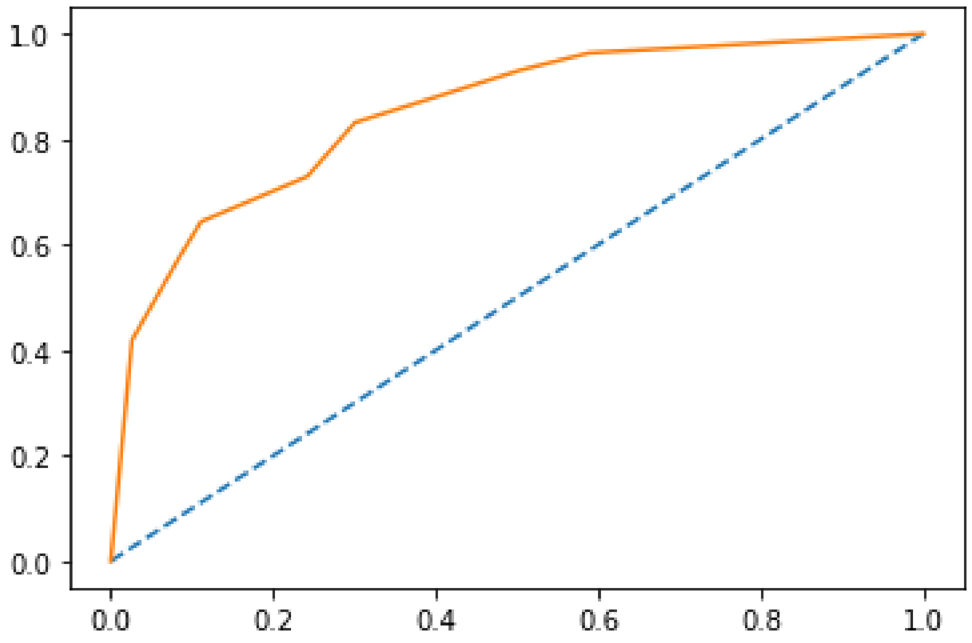


Figure 55. ADA Boosting: Train Confusion Matrix

```
array([[121, 186],
       [ 32, 722]], dtype=int64)
```

Figure 56. ADA Boosting: Train Classification Report

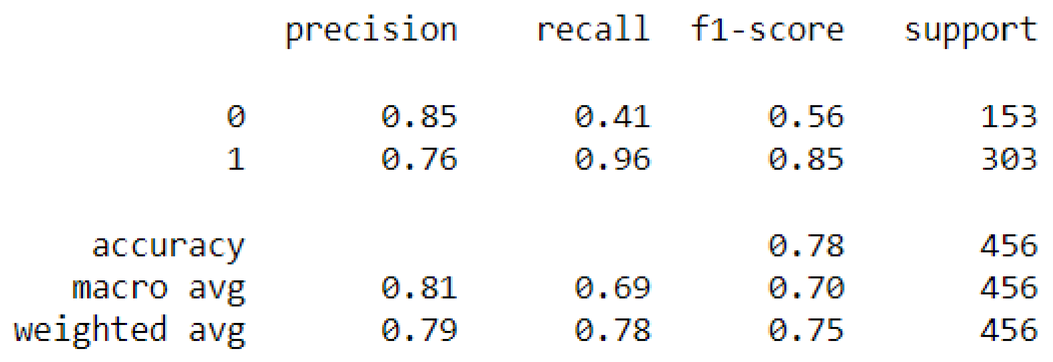
```
              precision    recall  f1-score   support

           0       0.79      0.39      0.53       307
           1       0.80      0.96      0.87       754

    accuracy                           0.79      1061
   macro avg       0.79      0.68      0.70      1061
weighted avg       0.79      0.79      0.77      1061
```

Figure 57. ADA Boosting: Test Confusion Matrix

```
array([[ 63,  90],
       [ 11, 292]], dtype=int64)
```

Back to Top

Figure 58. ADA Boosting: Test Classification Report



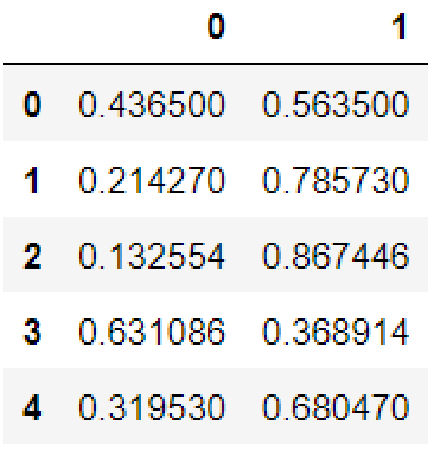
```
              precision    recall  f1-score   support

           0       0.85      0.41      0.56       153
           1       0.76      0.96      0.85       303

    accuracy                           0.78       456
   macro avg       0.81      0.69      0.70       456
weighted avg       0.79      0.78      0.75       456
```

1.1.8.3 Gradient Boosting

Gradient Boosting: (Please refer to the Jupyter notebook)

- The GradientBoostingClassifier() function is fit on the X_train and y_train datasets
- The training and test datasets are then predicted
- The next step would be to get the predicted classes and probabilities:

Figure 59. Predicted Classes and Probabilities



	0	1
0	0.436500	0.563500
1	0.214270	0.785730
2	0.132554	0.867446
3	0.631086	0.368914
4	0.319530	0.680470

Back to Top

Figure 60. Gradient Boosting: Train AUC Curve

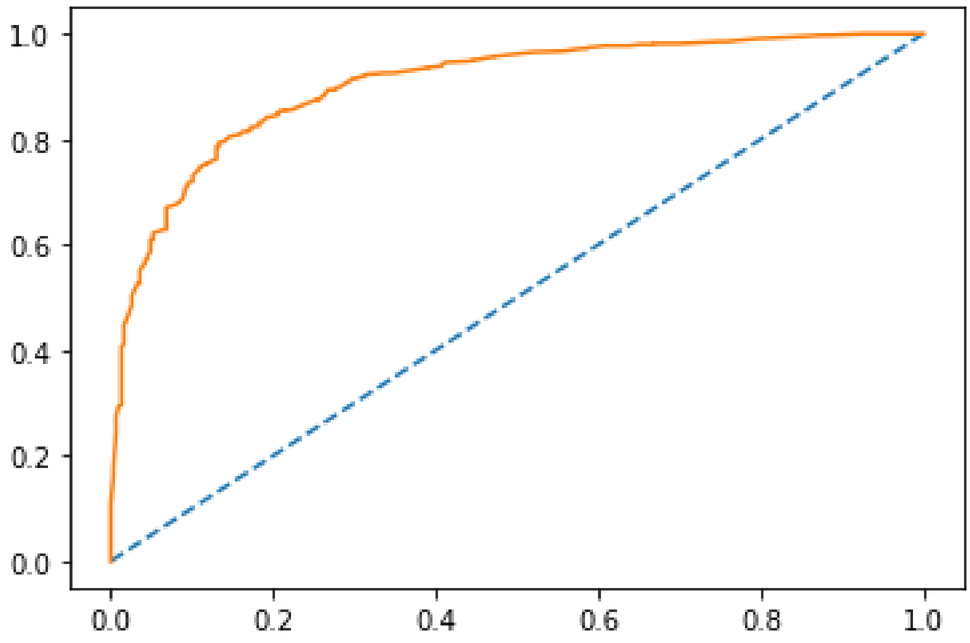
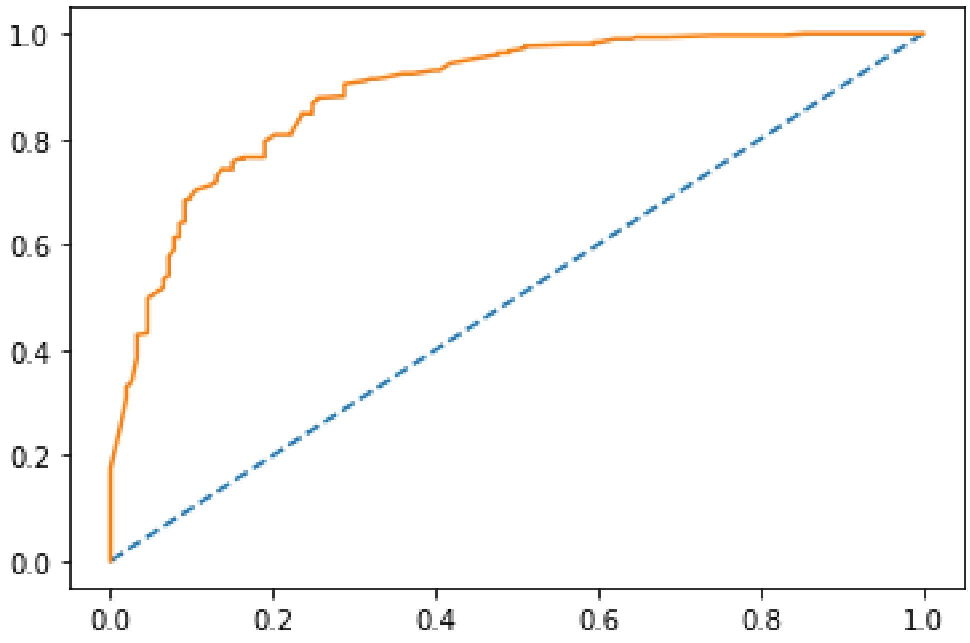
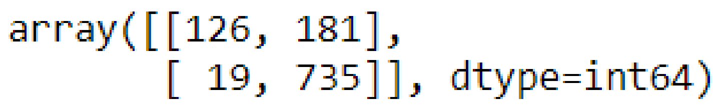


Figure 61. Gradient Boosting: Test AUC Curve



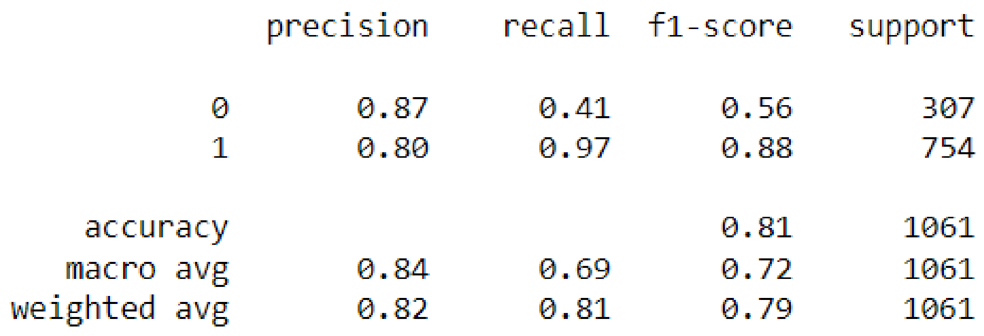
Back to Top

Figure 62. Gradient Boosting: Train Confusion Matrix



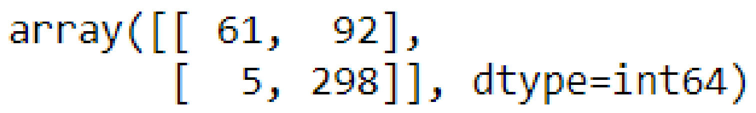
```
array([[126, 181],
       [ 19, 735]], dtype=int64)
```

Figure 63. Gradient Boosting: Train Classification Report



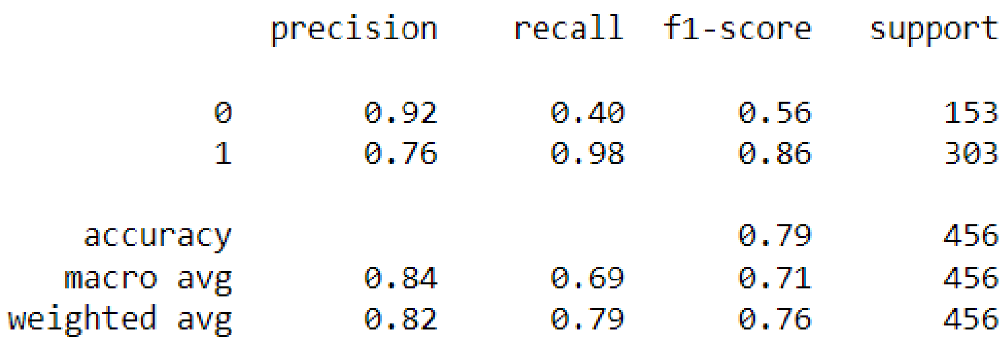
```
              precision    recall  f1-score   support

           0       0.87      0.41      0.56       307
           1       0.80      0.97      0.88       754

    accuracy                           0.81      1061
   macro avg       0.84      0.69      0.72      1061
weighted avg       0.82      0.81      0.79      1061
```

Figure 64. Gradient Boosting: Test Confusion Matrix



```
array([[ 61,  92],
       [  5, 298]], dtype=int64)
```

Figure 65. Gradient Boosting: Test Classification Report



```
              precision    recall  f1-score   support

           0       0.92      0.40      0.56       153
           1       0.76      0.98      0.86       303

    accuracy                           0.79       456
   macro avg       0.84      0.69      0.71       456
weighted avg       0.82      0.79      0.76       456
```

Table 4 Combined Model: Labour

	Logistic Regression Train	Logistic Regression Test	LDA Train	LDA Test	KNN Train	KNN Test	NB Train	NB Test	Bagging Train	Bagging Test	ADA Boosting Train	ADA Boosting Test	Grad Boosting Train	Grad Boosting Test
Accuracy	0.83	0.83	0.83	0.83	0.86	0.82	0.83	0.82	0.97	0.83	0.79	0.78	0.81	0.79
Recall	0.91	0.88	0.91	0.88	0.93	0.91	0.89	0.87	0.99	0.90	0.96	0.96	0.97	0.98
Precision	0.86	0.87	0.86	0.86	0.88	0.84	0.88	0.87	0.96	0.85	0.80	0.76	0.80	0.76
F1 Score	0.89	0.87	0.89	0.87	0.90	0.87	0.88	0.87	0.98	0.88	0.87	0.85	0.88	0.86

Table 5 Combined Model: Conservative

	Logistic Regression Train	Logistic Regression Test	LDA Train	LDA Test	KNN Train	KNN Test	NB Train	NB Test	Bagging Train	Bagging Test	ADA Boosting Train	ADA Boosting Test	Grad Boosting Train	Grad Boosting Test
Accuracy	0.83	0.83	0.83	0.83	0.86	0.82	0.83	0.82	0.97	0.83	0.79	0.78	0.81	0.79
Recall	0.64	0.73	0.65	0.73	0.68	0.65	0.69	0.73	0.90	0.68	0.39	0.41	0.41	0.40
Precision	0.75	0.76	0.74	0.76	0.80	0.78	0.72	0.74	0.99	0.78	0.79	0.85	0.87	0.92
F1 Score	0.69	0.74	0.69	0.74	0.74	0.71	0.71	0.73	0.94	0.73	0.53	0.56	0.56	0.56

1.1.9 Model Tuning with Hyper-parameters

1.1.9.1 Logistic Regression

Logistic Regression: (Please refer to the Jupyter notebook)

- The LogisticRegression() function is fit on the X_train and y_train datasets
- Hyper-parameters:
  - Penalty: ['l2','none'],
  - Solver: ['sag','lbfgs', 'liblinear'],
  - Tolerance: [0.0001,0.00001]
- Selected hyper-parameters using GridSearchCV:
  - Penalty: ['l2'],
  - Solver: ['lbfgs', 'liblinear'],
  - Tolerance: [0.0001]
- The training and test datasets are then predicted

Back to Top

Figure 66. Logistic Regression (Post Tuning): Train AUC Curve

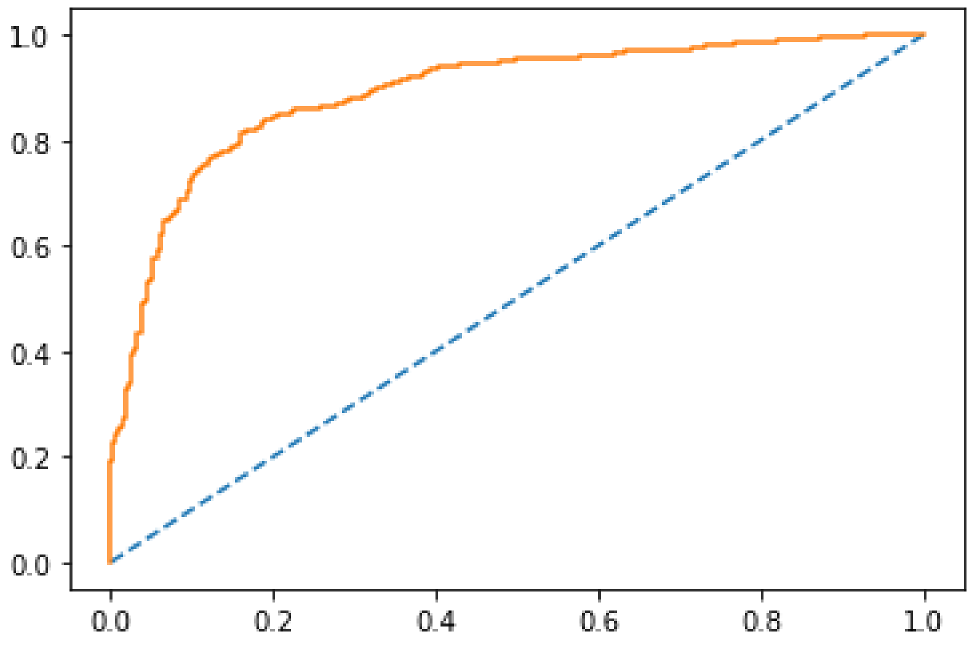
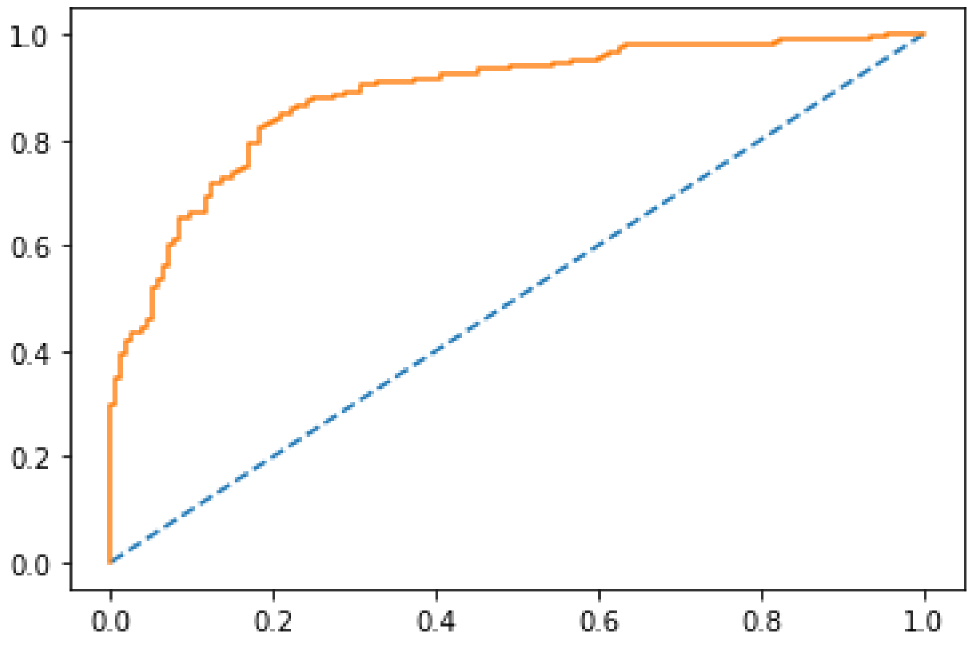
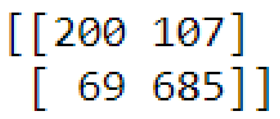


Figure 67. Logistic Regression (Post Tuning): Test AUC Curve



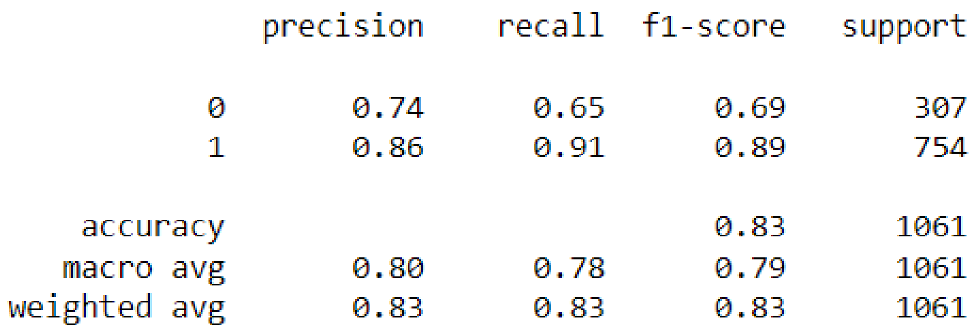
Back to Top

Figure 68. Logistic Regression (Post Tuning): Train Confusion Matrix



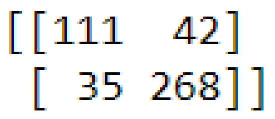
```
[[200 107]
 [ 69 685]]
```

Figure 69. Logistic Regression (Post Tuning): Train Classification Report



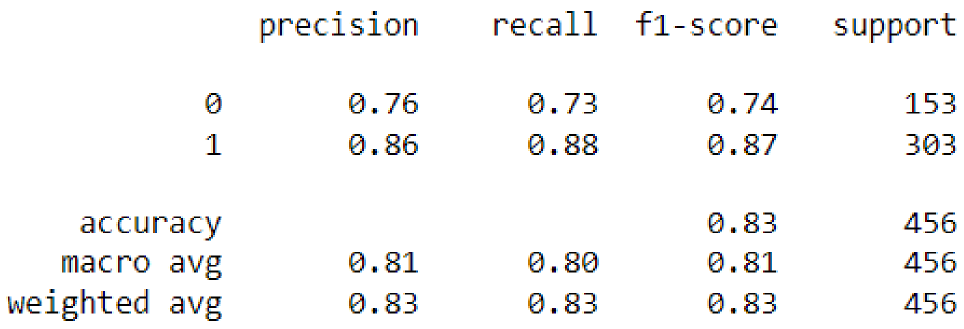
```
              precision    recall  f1-score   support

           0       0.74      0.65      0.69       307
           1       0.86      0.91      0.89       754

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

Figure 70. Logistic Regression (Post Tuning): Test Confusion Matrix



```
[[111  42]
 [ 35 268]]
```

Figure 71. Logistic Regression (Post Tuning): Test Classification Report



```
              precision    recall  f1-score   support

           0       0.76      0.73      0.74       153
           1       0.86      0.88      0.87       303

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

1.1.9.2 LDA

LDA: (Please refer to the Jupyter notebook)

- The LinearDiscriminantAnalysis() function is fit on the X_train and y_train datasets
- Hyper-parameters:
  - Solver: ['svd', 'lsqr', 'eigen']
- Selected hyper-parameters using GridSearchCV:
  - Solver: ['svd']
- The training and test datasets are then predicted

Back to Top

Figure 72. LDA (Post Tuning): Train AUC Curve

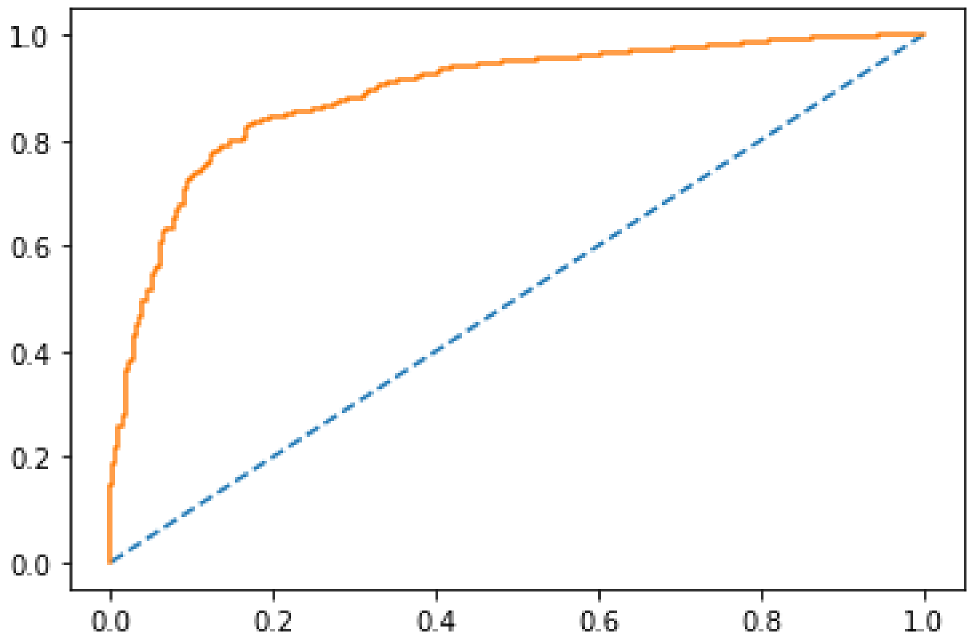


Figure 73. LDA (Post Tuning): Test AUC Curve

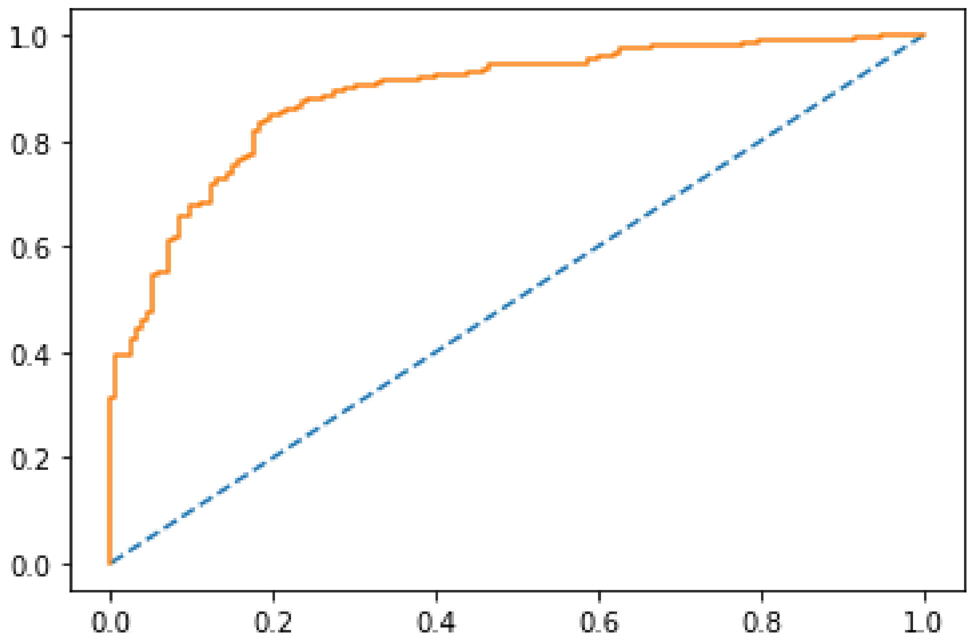
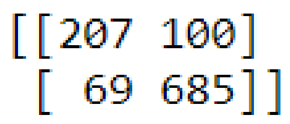
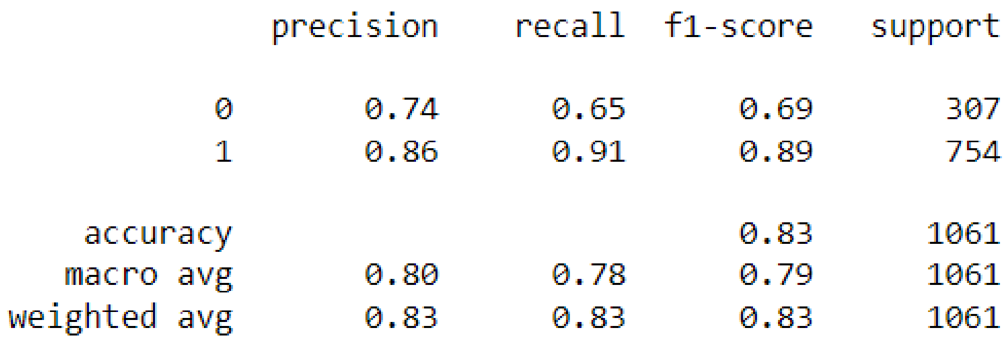


Figure 74. LDA (Post Tuning): Train Confusion Matrix



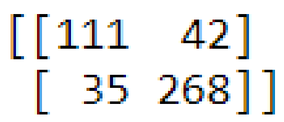
```
[[207 100]
 [ 69 685]]
```

Figure 75. LDA (Post Tuning): Train Classification Report



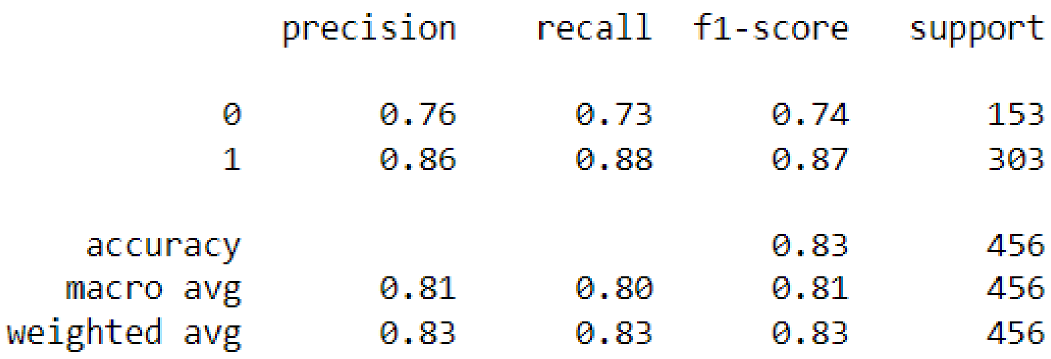
```
              precision    recall  f1-score   support

           0       0.74      0.65      0.69       307
           1       0.86      0.91      0.89       754

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

Figure 76. LDA (Post Tuning): Test Confusion Matrix



```
[[111  42]
 [ 35 268]]
```

Figure 77. LDA (Post Tuning): Test Classification Report



```
              precision    recall  f1-score   support

           0       0.76      0.73      0.74       153
           1       0.86      0.88      0.87       303

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

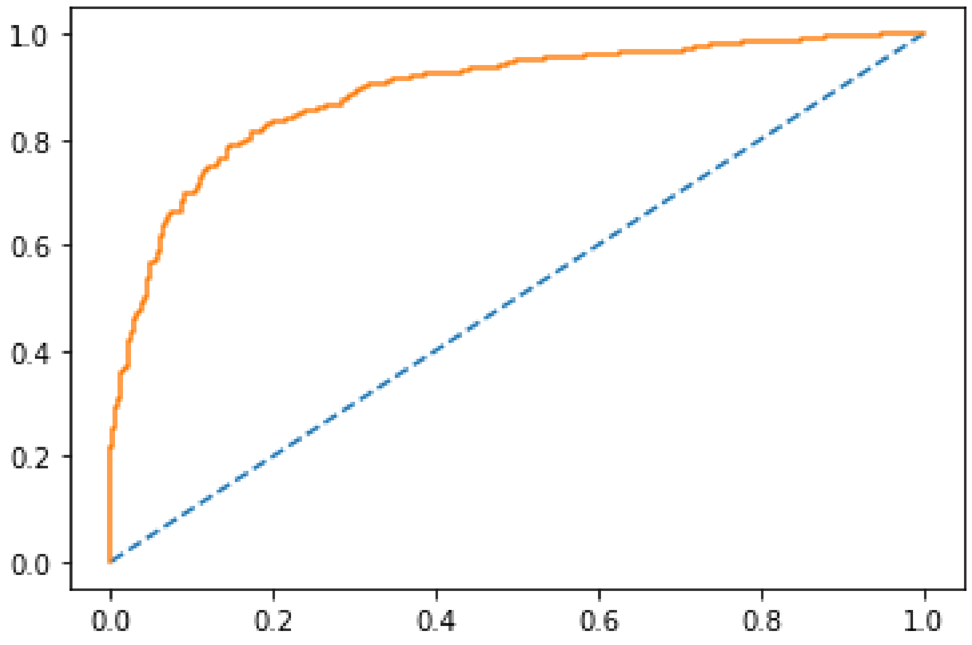
Back to Top

1.1.9.3 Naive Bayes

Naive Bayes: (Please refer to the Jupyter notebook)

- The GaussianNB() function is fit on the X_train and y_train datasets
- Hyper-parameters:
    - 'var_smoothing': np.logspace(0,-9, num=100)
- Selected hyper-parameters using GridSearchCV:
    - 'var_smoothing': 0.0003511191734215131
- The training and test datasets are then predicted

Figure 78. Naive Bayes (Post Tuning): Train AUC Curve



Back to Top

Figure 79. Naive Bayes (Post Tuning): Test AUC Curve

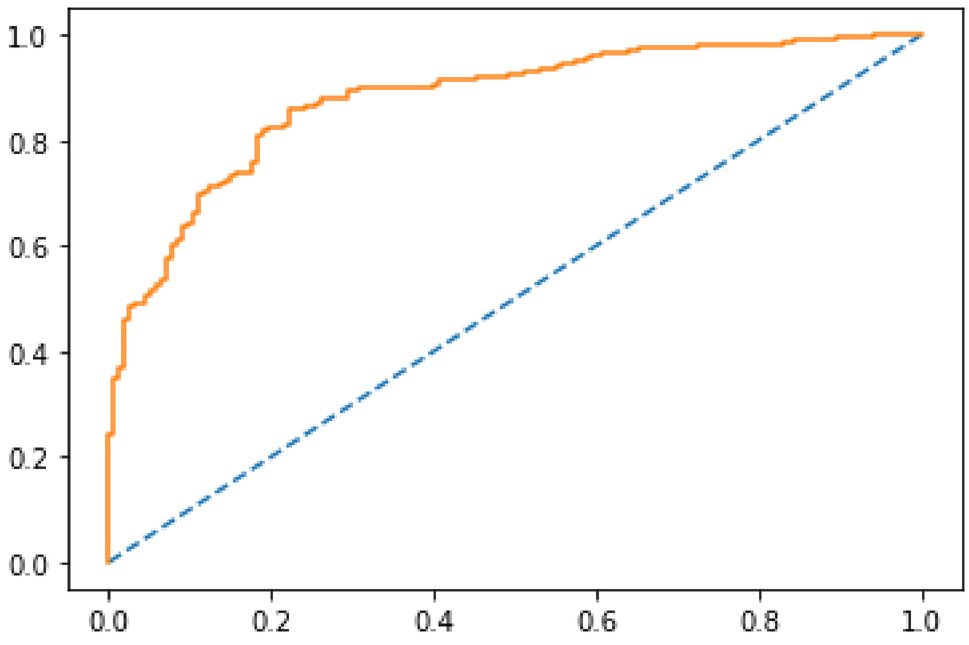


Figure 80. Naive Bayes (Post Tuning): Train Confusion Matrix

```
[[212  95]
 [ 77 677]]
```

Figure 81. Naive Bayes (Post Tuning): Train Classification Report

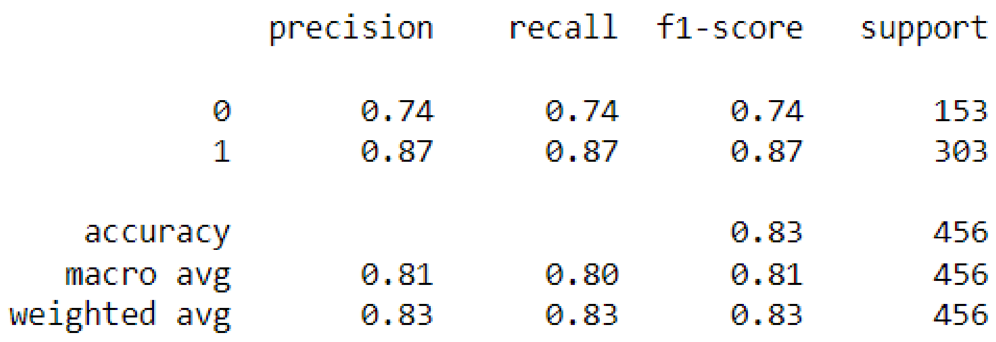
```
              precision    recall  f1-score   support

           0       0.73      0.69      0.71       307
           1       0.88      0.90      0.89       754

    accuracy                           0.84      1061
   macro avg       0.81      0.79      0.80      1061
weighted avg       0.84      0.84      0.84      1061
```

Figure 82. Naive Bayes (Post Tuning): Test Confusion Matrix

```
[[113  40]
 [ 39 264]]
```

Back to Top

Figure 83. Naive Bayes (Post Tuning): Test Classification Report



```
              precision    recall  f1-score   support

           0       0.74      0.74      0.74       153
           1       0.87      0.87      0.87       303

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

1.1.9.4 KNN

KNN: (Please refer to the Jupyter notebook)

- The KNeighborClassifier() function is fit on the X_train and y_train datasets
- Hyper-parameters:
    - 'n_neighbors': [5,6,7,8,9,10],
    - 'Leaf_size': [1,2,3,5],
    - 'weights': ['uniform', 'distance'],
    - 'algorithm': ['auto', 'ball_tree','kd_tree','brute'],
    - 'n_jobs': [-1]
- Selected hyper-parameters using GridSearchCV:
    - 'algorithm': 'auto',
    - 'leaf_size': 5,
    - 'n_jobs': -1,
    - 'n_neighbors': 9,
    - 'weights': 'distance'
- The training and test datasets are then predicted

Back to Top

Figure 84. KNN (Post Tuning): Train AUC Curve

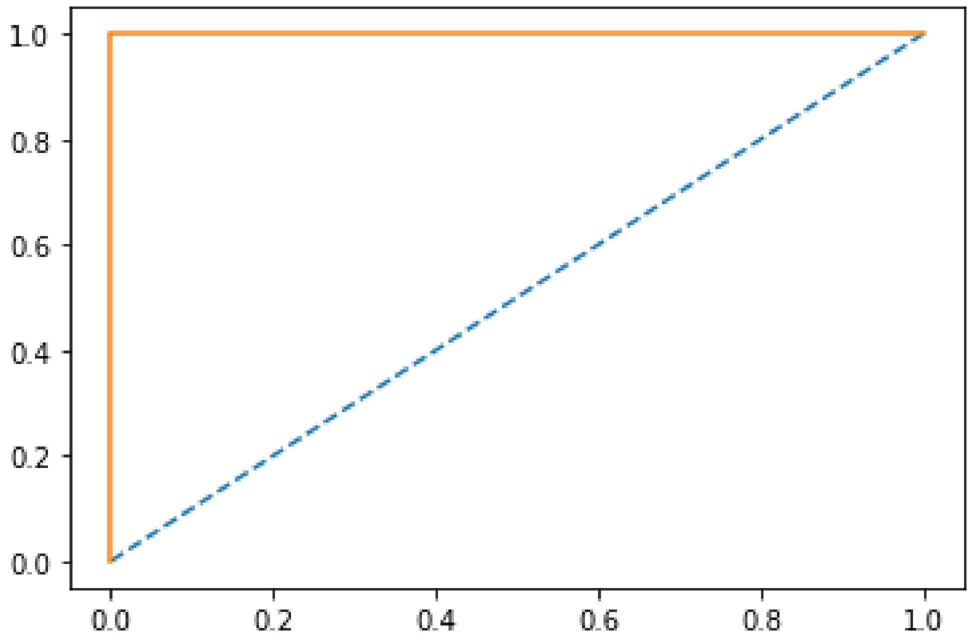


Figure 85. KNN (Post Tuning): Test AUC Curve

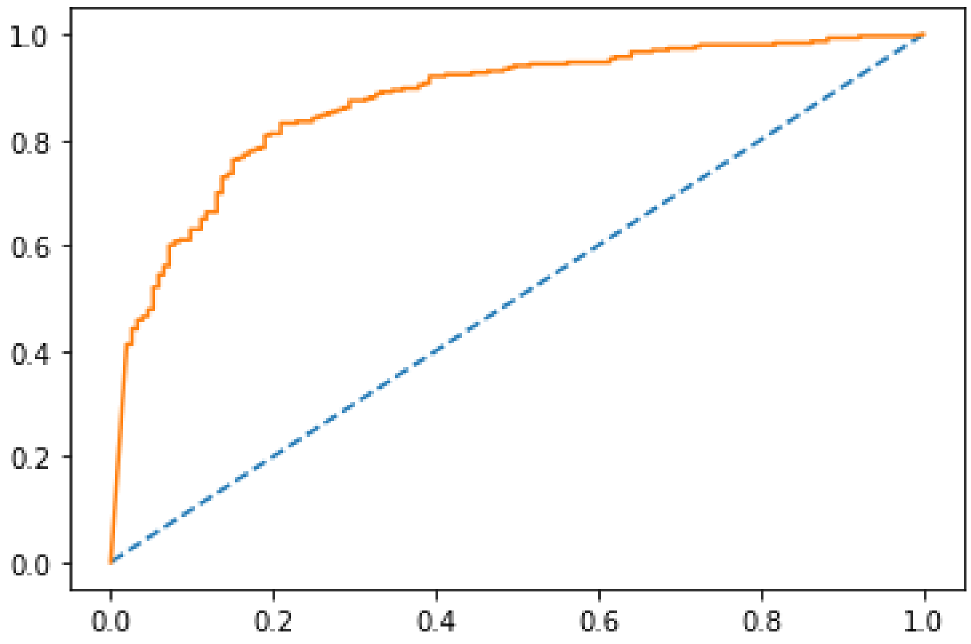
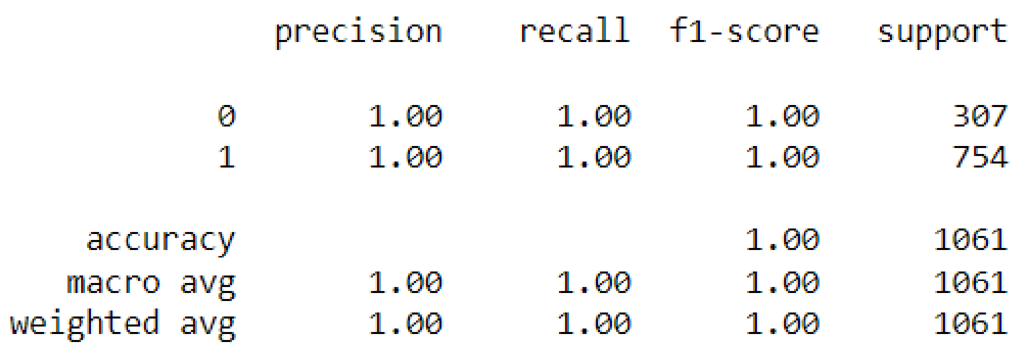


Figure 86. KNN (Post Tuning): Train Confusion Matrix

```
[[307   0]
 [  0 754]]
```

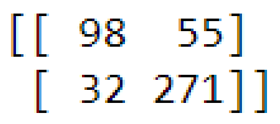
Back to Top

Figure 87. KNN (Post Tuning): Train Classification Report



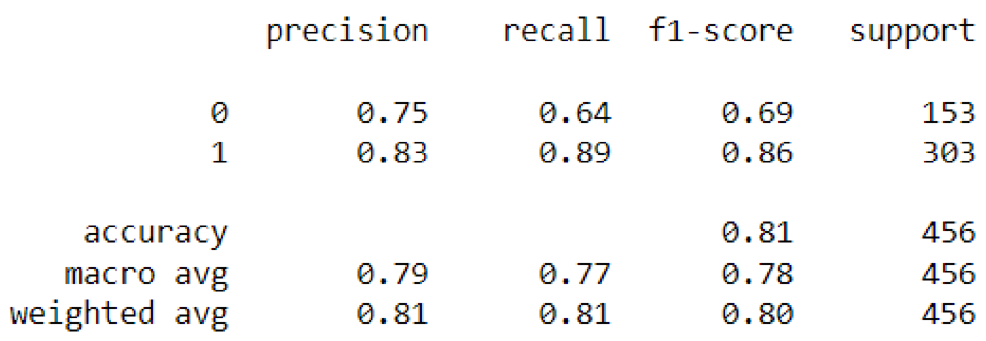
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       307
           1       1.00      1.00      1.00       754

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

Figure 88. KNN (Post Tuning): Test Confusion Matrix



```
[[ 98  55]
 [ 32 271]]
```

Figure 89. KNN (Post Tuning): Test Classification Report



```
              precision    recall  f1-score   support

           0       0.75      0.64      0.69       153
           1       0.83      0.89      0.86       303

    accuracy                           0.81       456
   macro avg       0.79      0.77      0.78       456
weighted avg       0.81      0.81      0.80       456
```

1.1.10 Best Model:

After tuning the Logistic Regression, LDA, KNN, and Naïve Bayes models, we can see that model constructed through using gradient boosting classifier performs much better. With better recall, accuracy, and f1 score, we can come to a conclusion that gradient boosting model is the best model for the given dataframe.

Back to Top

1.1.11 Based on these predictions, what are the insights?

Through the various data exploration techniques, we have obtained a fair idea about the characteristics of the people who opt to vote for the Labour Party and the Conservative Party. The characteristics can be summarized in the following table:

Particulars	Labour Party	Conservative Party
Age	Younger people seem to prefer the Labour party	Older people seem to prefer the Conservative Party
National Economic Condition	People having an optimistic opinion seem to choose the Labour Party	People having a pessimistic opinion seem to choose the Conservative Party
Household Econmic Condition	People having an optimistic opinion seem to choose the Labour Party	People having a pessimistic opinion seem to choose the Conservative Party
Eurosceptic Sentiments	People who support close ties between Britain and the EU seem to prefer voting for the Labour party	People having largely 'Eurosceptic' sentiments have voted for the Conservative Party
Political Knowledge	Most of the Labour Party voters do not seem to be well-informed; can be assumed to be relatively naïve about their party's stand on European integration	Most of the Conservative Party voters seem to be well-informed about the party's stance on European integration and they support its ideologies.

- We have also observed that as people's political knowledge increases, the Eurosceptic sentiment also increases. This could possibly hint at the negative aspects of close ties between Britain and the EU. Apart from using the data for exit poll prediction purposes, the news channel can also focus on introducing programs/debates on the implications of Britain's exit from the EU.
- The various models which were built (Logistic Regression, LDA, Naïve Bayes, KNN, Bagging, and Boosting) predicted whether a voter voted for the Labour Party or the Conservative Party on the basis of the user's characteristics and opinions on various macro-economic factors.
- These models can be extended to the larger population as they are observed to perform well on the sample data.
- A possible suggestion would be also to understand some more characteristics about the respondents such as the area in which they live, employment status, etc. so that we can get even more supporting factors that play a pivotal role in determining the preferred political party.

Back to Top

Chapter 2. Problem 2

2.1 Problem Statement

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

- President Franklin D. Roosevelt in 1941
- President John F. Kennedy in 1961
- President Richard Nixon in 1973

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

- **Roosevelt's speech:** (Please refer to the Jupyter notebook)
    - Number of sentences: 38
    - Number of words: 1,360
    - Number of characters (including spaces): 7,496

- **Kennedy's speech:** (Please refer to the Jupyter notebook)
    - Number of sentences: 27
    - Number of words: 1,390
    - Number of characters (including spaces): 7,565

- **Nixon's speech:** (Please refer to the Jupyter notebook)
    - Number of sentences: 51
    - Number of words: 1,819
    - Number of characters (including spaces): 9,890

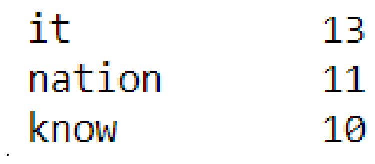
2.1.2 Remove all the stopwords from all three speeches.

- After downloading the "stopwords" package and importing it, we select the English stop words
- The next step would be to check how many stop words are there in each speech. (Please refer to the Jupyter notebook)
    - Roosevelt's speech: 632 stop words
    - Kennedy's speech: 618 stop words
    - Roosevelt's speech: 899 stop words

Back to Top

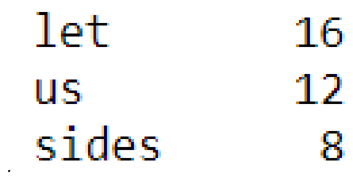
2.1.3 Which word occurs the most number of times in his inaugural address for each president? Mention the top three words. (after removing the stopwords)

Figure 90. Roosevelt Speech: Most Common Words



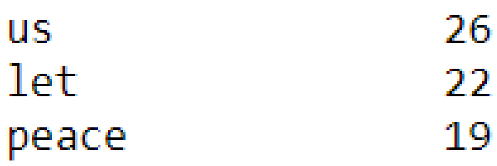
```
it          13
nation      11
know        10
```

Figure 91. Kennedy Speech: Most Common Words



```
let        16
us         12
sides       8
```

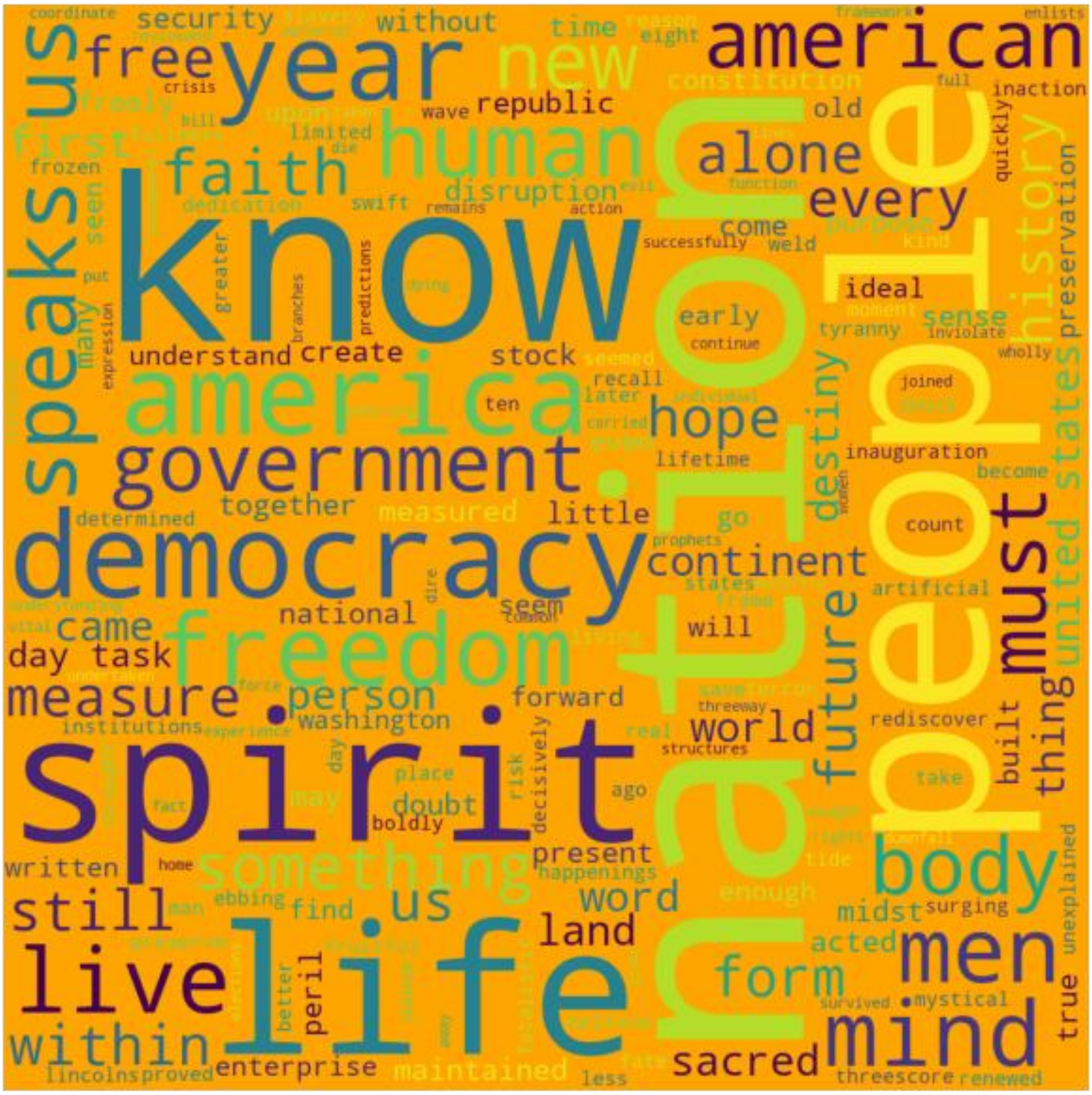
Figure 92. Nixon Speech: Most Common Words



```
us                 26
let                22
peace              19
```

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

Figure 93. Roosevelt Speech: WorldCloud

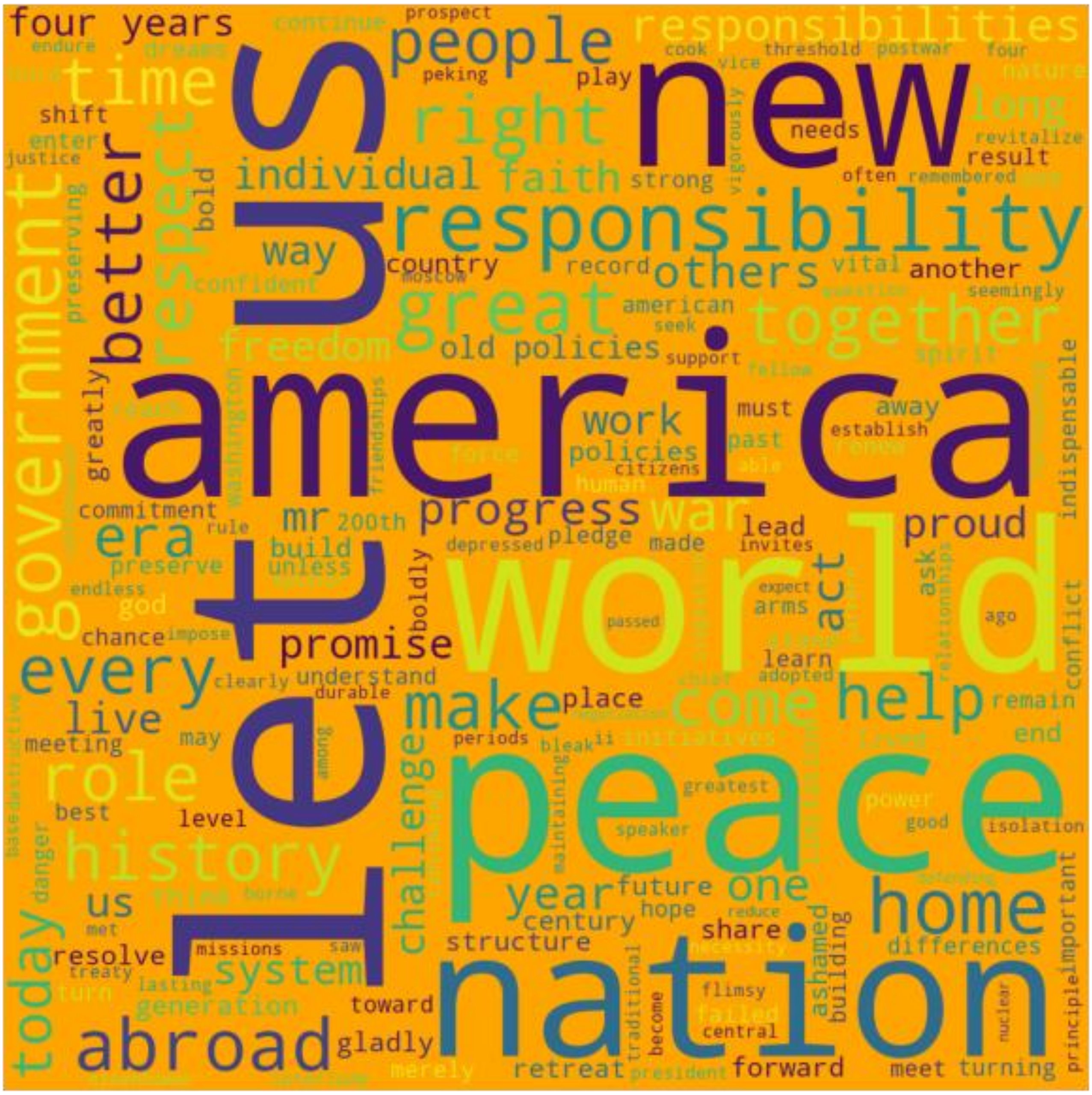


Back to Top

Figure 94. **Kennedy Speech: WorldCloud**

Back to Top

Figure 95. Nixon Speech: WorldCloud



Back to Top

The End

Back to Top