MACHINE LEARNING PROJECT BUSINESS REPORT

PG-DSBA

Written by

Priyamvada Singh

Dated: **06-04-2023**

(Format: dd-mm-yyyy)

Table of Contents

Table of Figures

Problem 1

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

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

- Basic analysis on the given data set:

i. The first five rows of the dataset:

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

ii. The last five rows of the data set:

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

iii. Dropping the 'unnamed' column:

	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

iv. Total number of rows and columns (features) present:
   - There are 1525 rows and 9 columns in the dataset.

v. Datatype of each feature, number of null values, duplicated records:

- *There are seven int64, and two object data type variables.*
- *That means, **7 variables are numeric while 2 are non-numeric**.*
- *However, only 1 numeric variable (age) is continuous variable while all the others are categorical but ordinal.*

vi. The following is the description of each numeric variable:

	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

- *The above information provides the mean, standard deviation, minimum, 25%, 50% (median), 75% and maximum data point values for each variable.*

vii. There are 8 duplicate values found in the data set which I dropped. Finally, there are 1517 rows and 9 columns.

viii. Inferences summary:

- No null values found.
- 8 duplicates found and dropped.
- No bad values found including spelling errors or letter case problems.
- There were 9 columns and 1525 rows in the dataset but came down to 1517 rows after dropping duplicates.
- There are two 'object' and seven 'int64' datatype variables in the dataset.
- There are two party choices, namely 'Labour' and 'Conservative', with majority dataset records belonging to 'Labour'. This is our class variable.
- In the gender category, two genders, namely 'female' and 'male' appear in the dataset, with majority records belonging to 'female'.
- The age of the records varies from 24 to 93, inclusive.
- All the other columns are ordinal with numeric categories, as given in the dataset dictionary.
- *Skewness:* All the numeric columns look well-balanced except 'age', `which is slightly skewed to the right, 'Europe' which is skewed to the left, and 'political_knowledge' which displays a significant skew to the right.

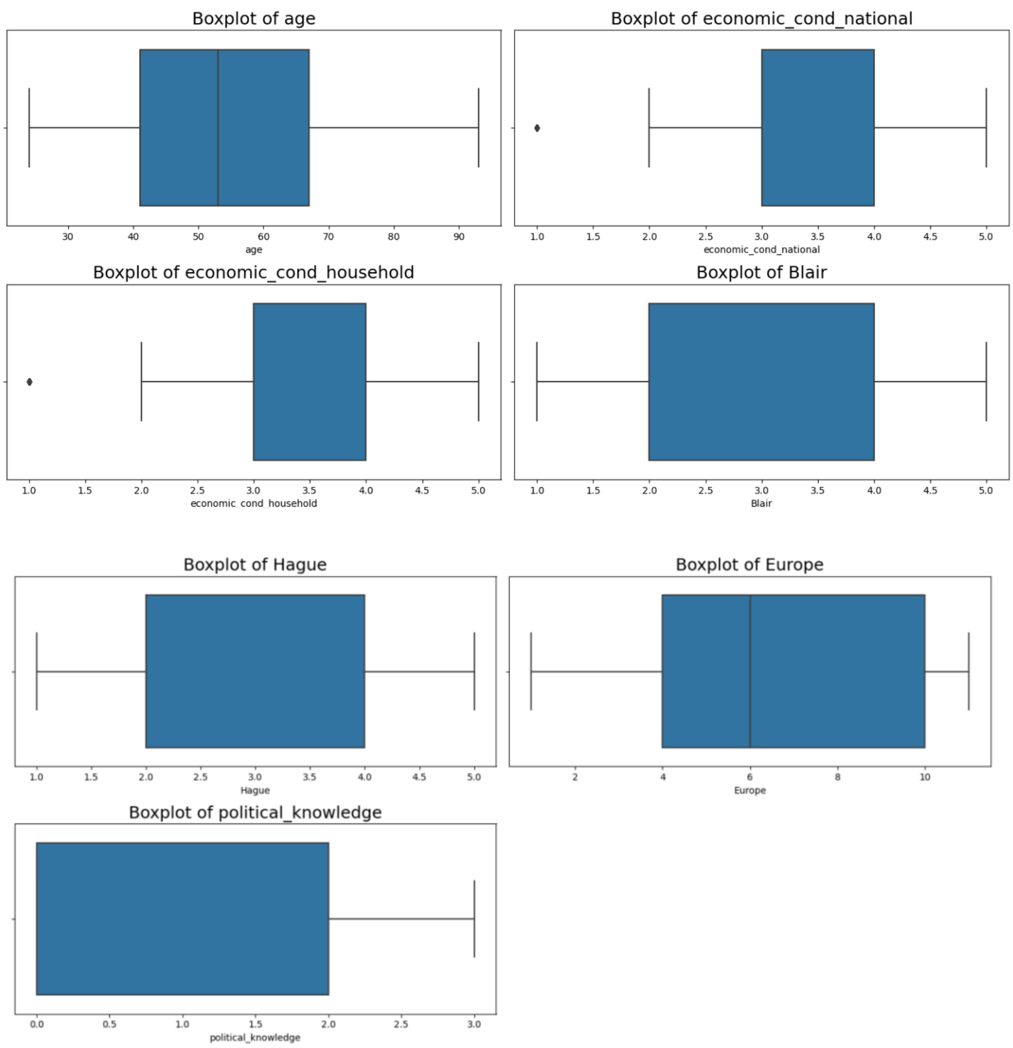
2. Perform EDA (Check the null values, Data types, shape, Univariate, bivariate analysis). Also check for outliers (4 pts). Interpret the inferences for each (3 pts) Distribution plots (histogram) or similar plots for the continuous columns. Box plots. Appropriate plots for categorical variables. Inferences on each plot. Outliers proportion should be discussed, and inferences from above used plots should be there. There is no restriction on how the learner wishes to implement this but the code should be able to represent the correct output and inferences should be logical and correct.

    i. There are two 'object' and seven 'int64' datatype variables in the dataset.
    ii. There were no null values in the dataset.

- EDA:

    i. Boxplots for all numeric variables to check the distribution and outliers:

Figure 1 - Boxplots



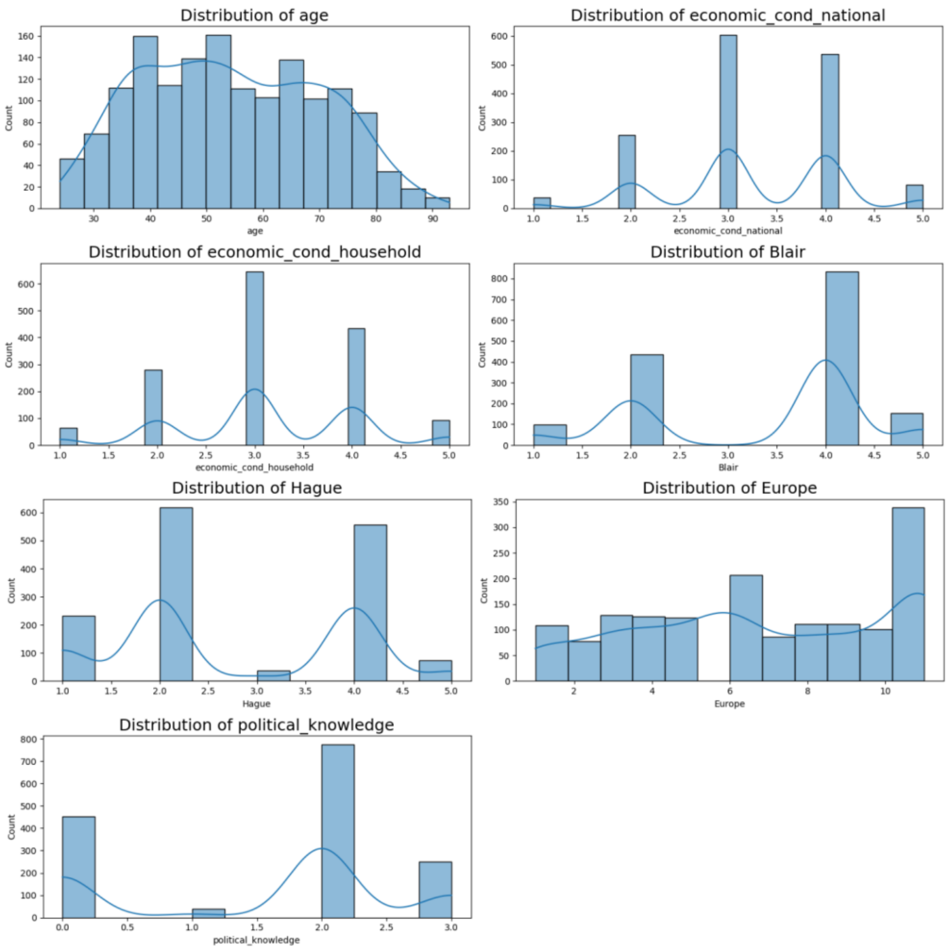
- Outliers:

i. Only two columns have outliers: 'economic_cond_household' and'economic_cond_national'. Both have outliers at the lower end.
ii. There are 102 records in total containing the outliers from 'economic_cond_national'(37) and 'economic_cond_household'(65). 15 records are common for both.
iii. Choosing to leave the outliers untreated as all the records look valid.
iv. Don't want to disturb or drop them.

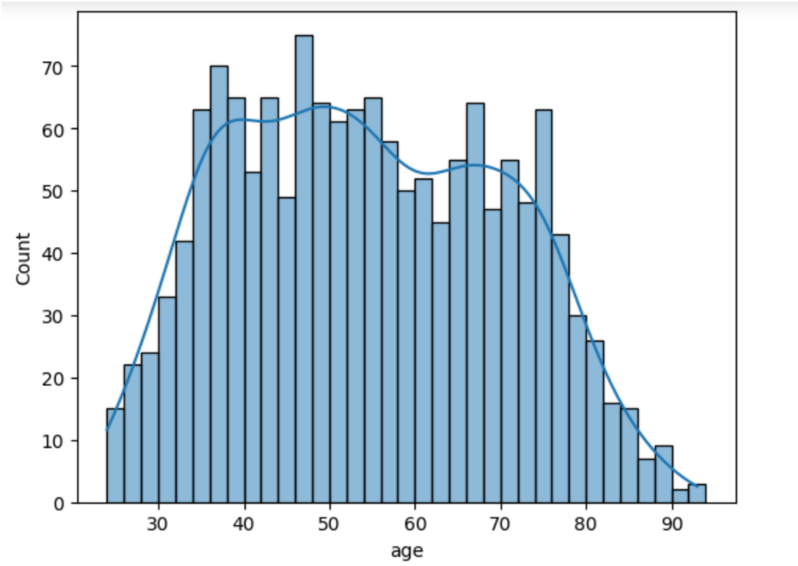
- Univariate Analysis:

i. Checking the distribution of each variable through histogram and kde:

Figure 2 - Histograms with KDE



ii. Analysing the 'age' variable again to check the distribution clearly and in detail:

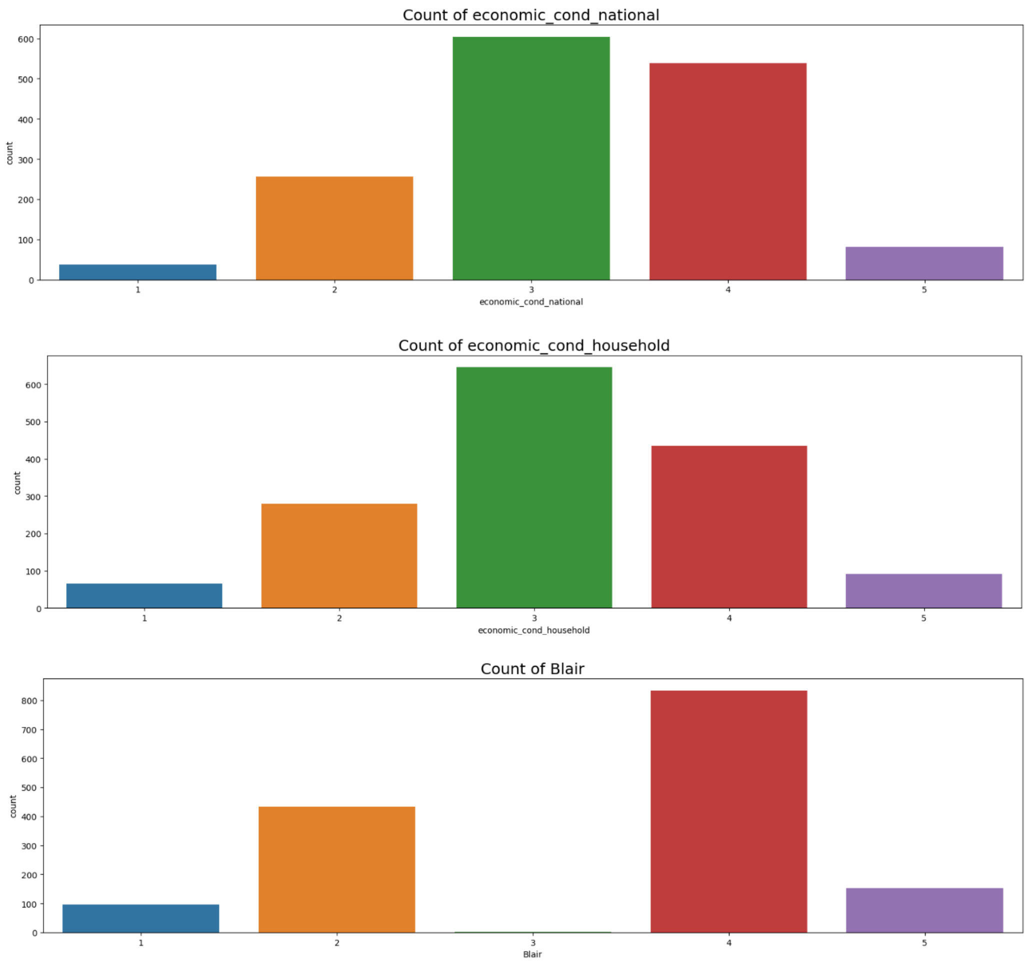


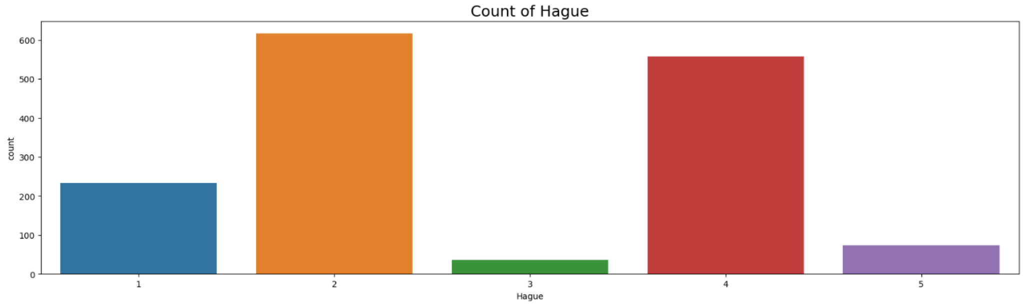
- *Inferences:*

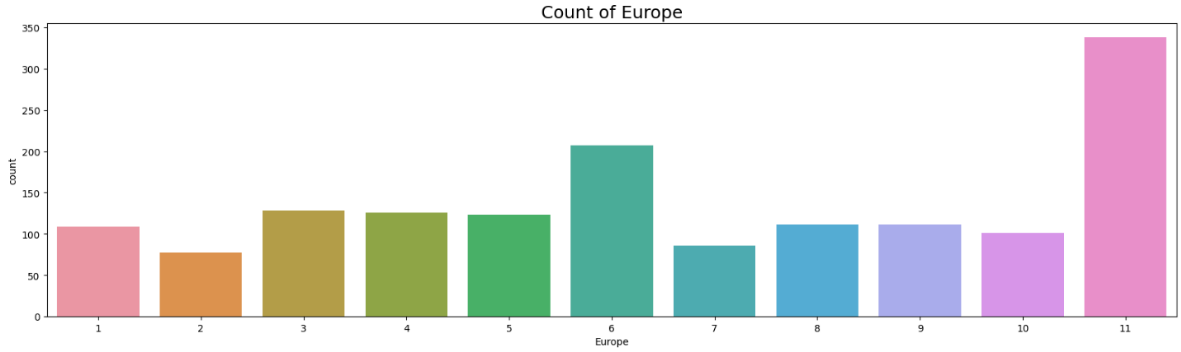
i. Ages from 34 to 76 seem to be among the most active records.

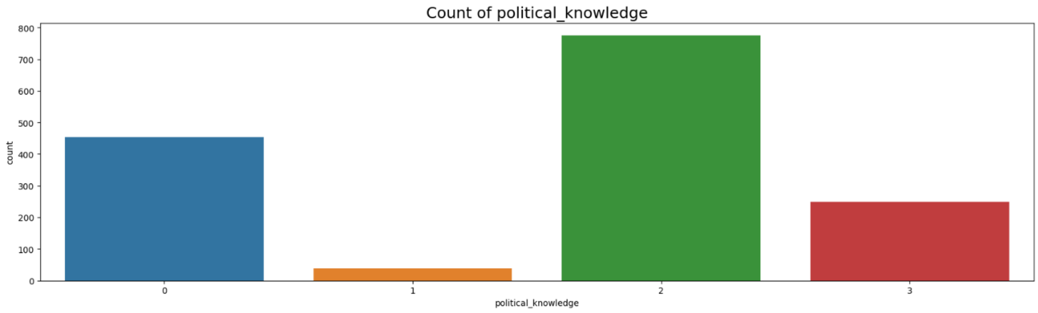
ii. Bar graphs for categorical variables to understand the distribution in depth:

Figure 3 - Bar Graphs









iii. Economic_cond_national and economic_cond_household have 3 as their most popular category

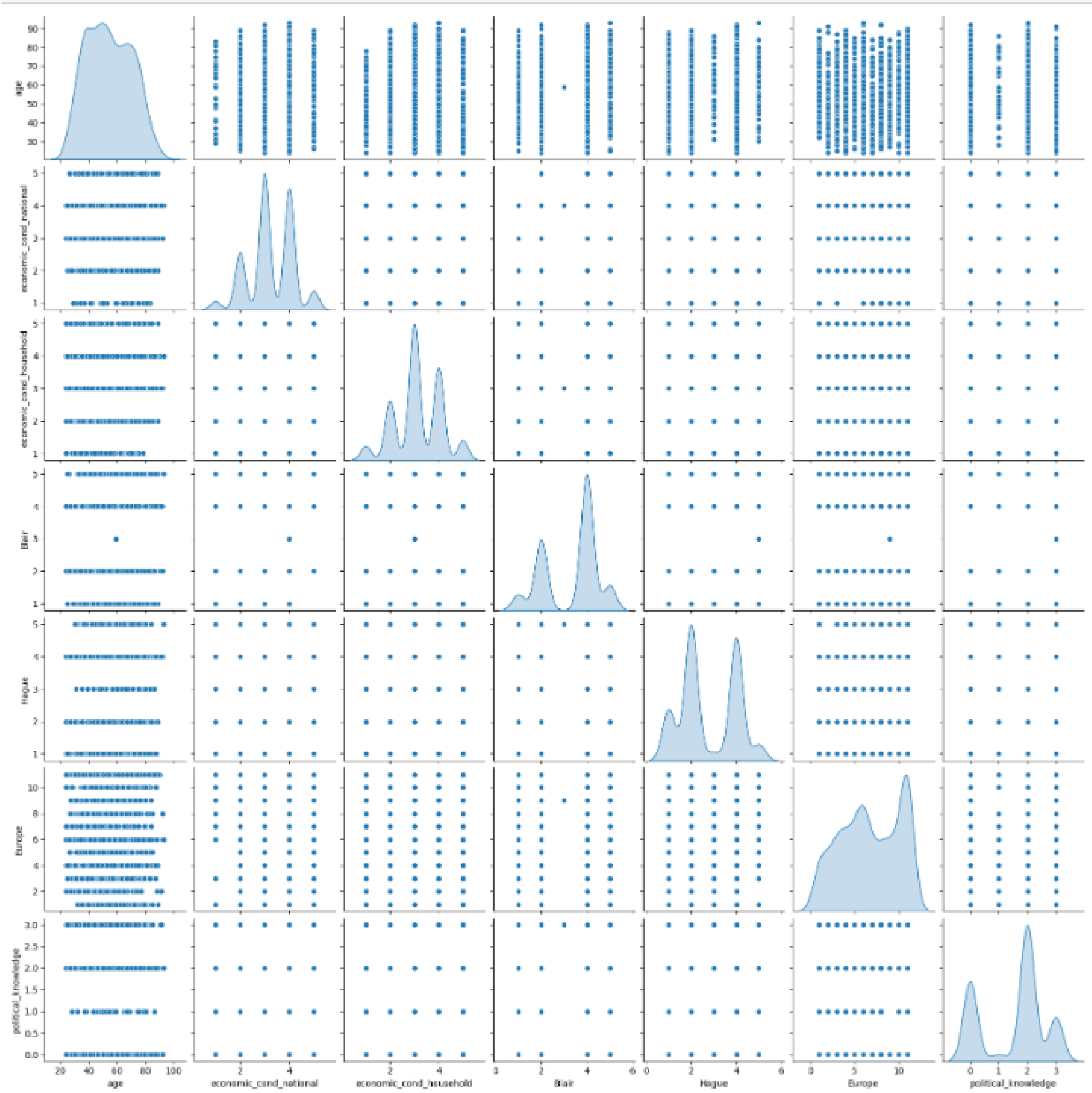
iv. 4 in 'Blair', 2 in 'Hague' and 11 in 'Europe' are the most popular categories.

v. Under 'political_knowledge', 2 is the most popular level.

- Bivariate Analysis:

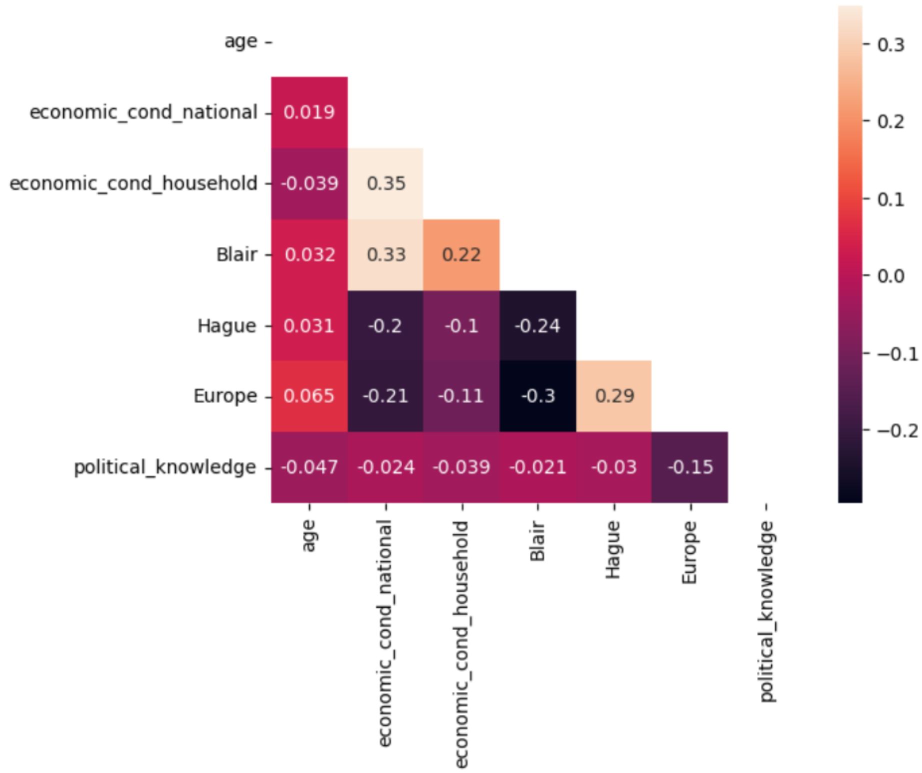
i. Constructing pairplot for all variables to check relationships between them:

Figure 4 - Pairplot



ii. Heatmap for all the features:

Figure 5 - Heatmap



iii. No significant correlation found in the heatmap.

iv. The following bar graph is for 'economic_cond_household' and 'economic_cond_national' with gender to check the trends and compare it for both genders.

Figure 6 - gender vs. economic_cond_household

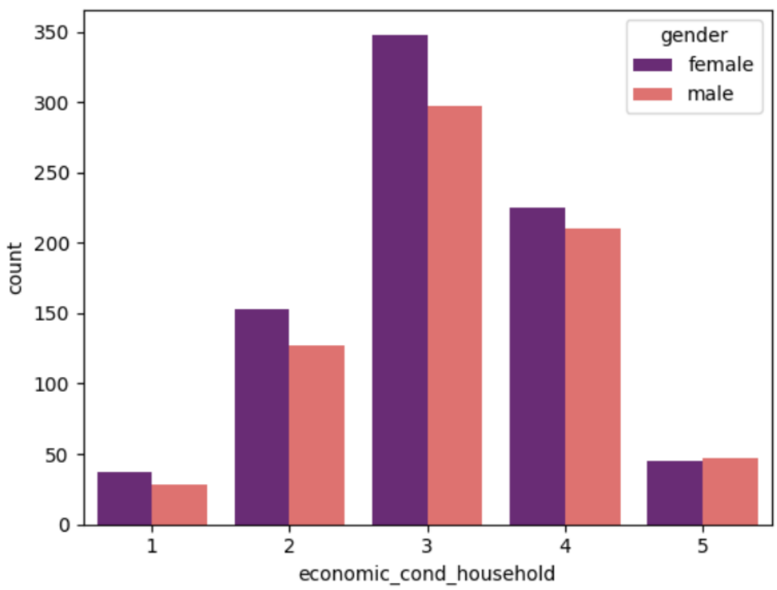
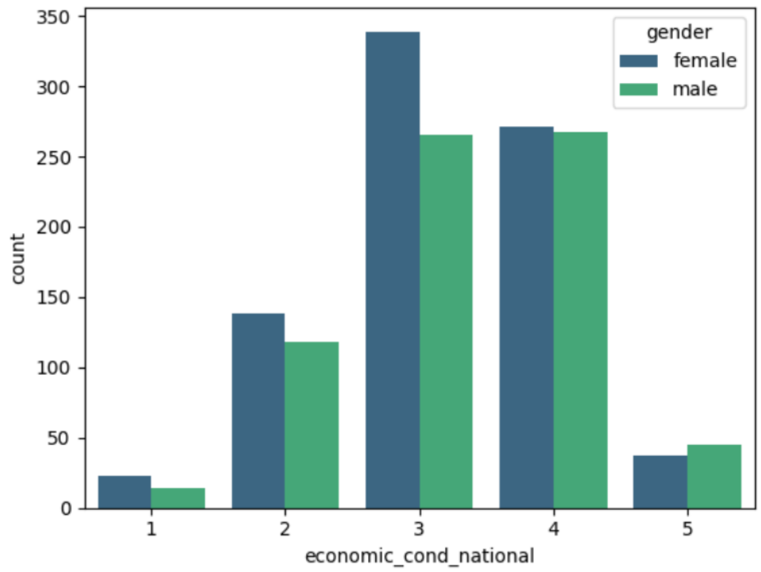


Figure 7 - gender vs. economic_cond_national

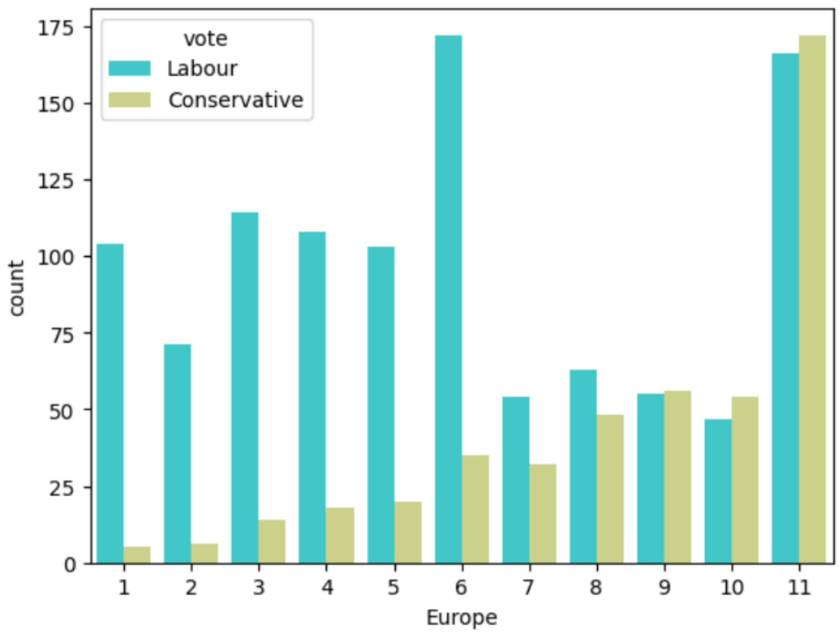


v. Male number is slightly higher under 5 category of the 'economic_cond_national' and 'economic_cond_household' variable.

vi. For both, 'economical_cond_national' and 'economic_cond_household', the trend for females and males is similar otherwise.

vii. Is there a difference in the severity of Eurosceptic sentiments between Labour and Conservative records?

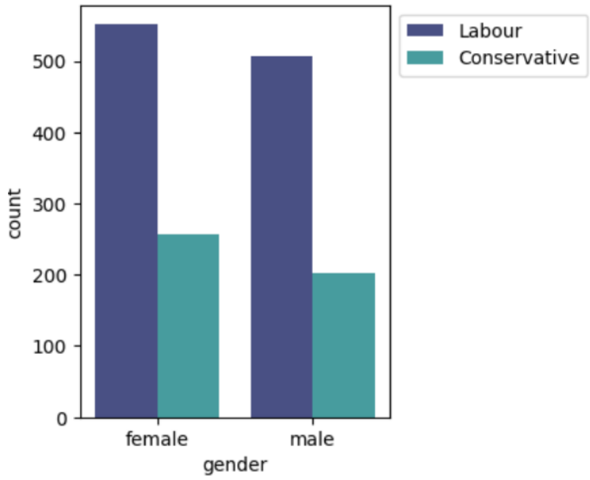
Figure 8 - Party Choice vs. Eurosceptic Sentiments



- Eurosceptic sentiments in Conservative records are *relatively* more severe than in Labour records with an exception, where 11 is the most popular rating under 'Europe' for both categories.

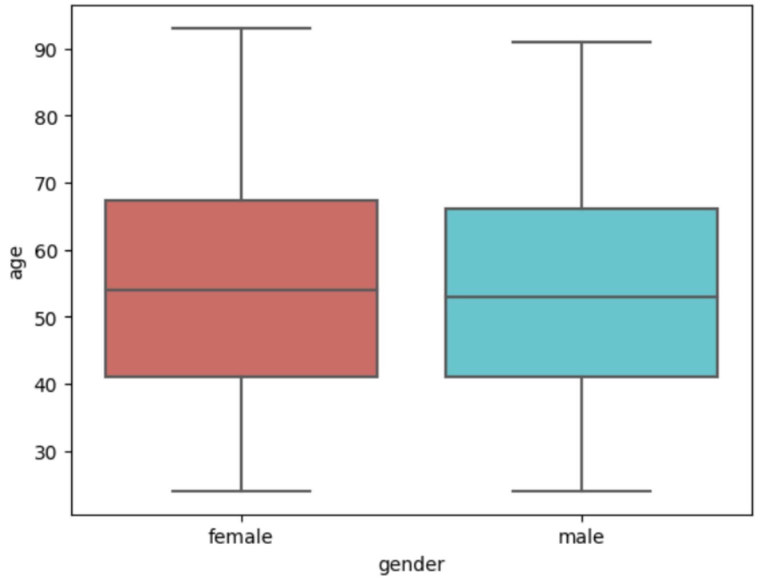
- If we only take a look till 10 in 'Europe' category, Labour and Conservative records are showing inverse trend.

Figure 9 - vote vs. gender



- Gender and party choice ('vote') do not appear to be correlated, as indicated by the above graph.

Figure 10 - age vs. gender



- No significant age distribution differences found based on gender.

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

   i. Changed the object variables – age and gender – to category:

```
<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   category
 1   age                      1517 non-null   int64
 2   economic_cond_national   1517 non-null   int64
 3   economic_cond_household  1517 non-null   int64
 4   Blair                    1517 non-null   int64
 5   Hague                    1517 non-null   int64
 6   Europe                   1517 non-null   int64
 7   political_knowledge      1517 non-null   int64
 8   gender                   1517 non-null   category
dtypes: category(2), int64(7)
memory usage: 130.3 KB
```

vi. Changed the object variables – age and gender – to category:

vii. Then one-hot encoded the gender category and changed the class variable 'vote' to binary, where Labour is 0 and Conservative is 1. Now, the dataset looks like below:

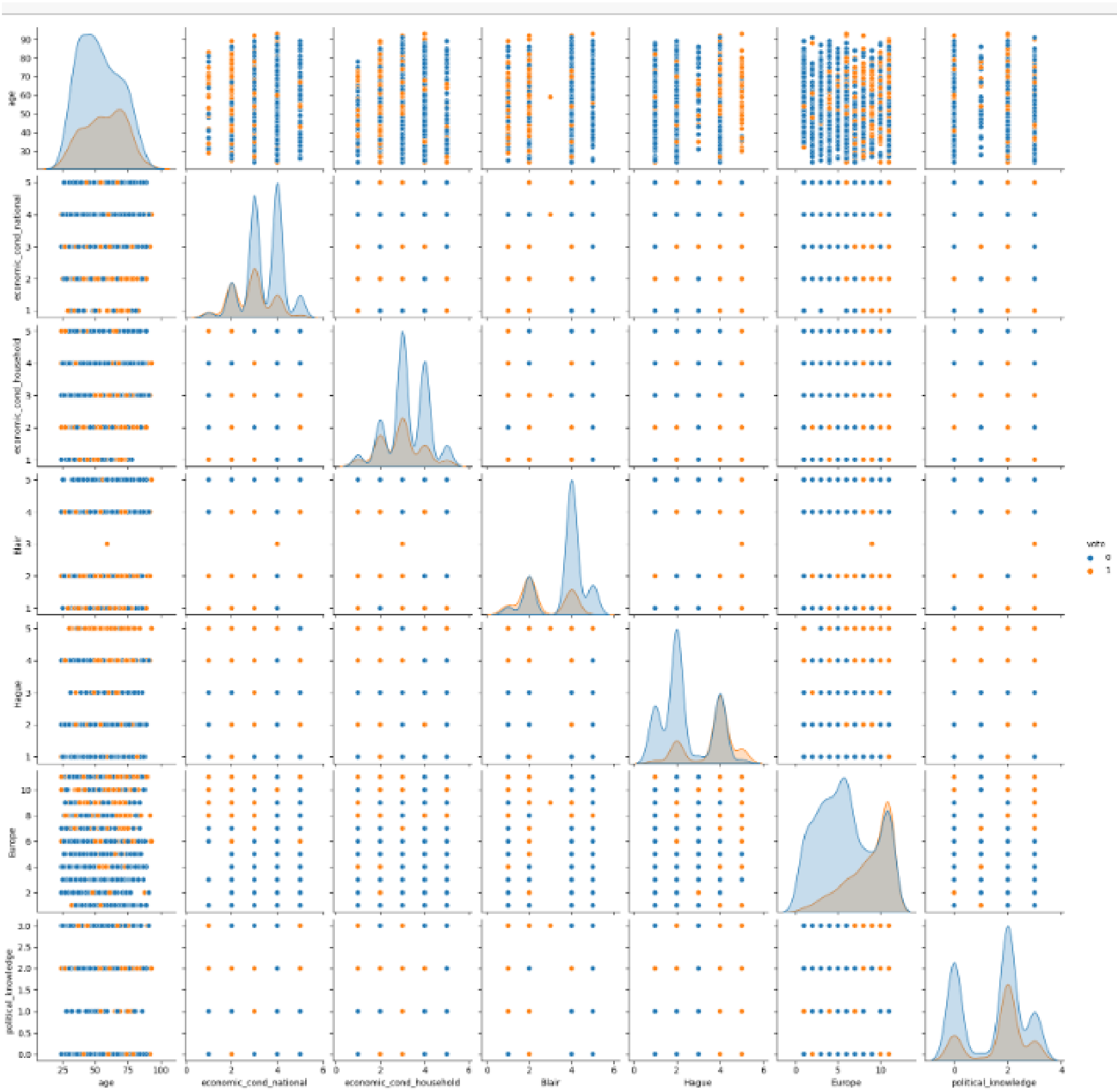
	vote	age	economic_cond_national	economic_cond_household	Blair	Hague	Europe	political_knowledge	gender_male
0	0	43	3	3	4	1	2	2	0
1	0	36	4	4	4	4	5	2	1
2	0	35	4	4	5	2	3	2	1
3	0	24	4	2	2	1	4	0	0
4	0	41	2	2	1	1	6	2	1

viii. The class variable 'vote' is imbalanced, hence, we need to keep this in mind while analysing the data and interpreting models.

```
0    0.69677
1    0.30323
Name: vote, dtype: float64
```

ix. Creating another data frame just to check the class overlap through the pairplot below:

Figure 11 - Pairplot II



Blue: 0 (Labour)
Orange: 1 (Conservative)

- *According to the above pairplot, most features are weak to poor predictors of the class variable.*

  - Scaling:

x. Scaling is recommended for Logistic Regression, LDA and KNN. This is because these models are sensitive to the scale of the input features and can make the model unstable if the features vary too much in their numerical values. Although, many features have close by numeric values, I will still go ahead and bring all the features to a standard scale.

xi. MinMaxScaler can be used to achieve this as the 'gender' variable has been encoded. This step will ensure that the binary values for 'gender' remain the same, along with scaling all the other variables (continuous and ordinal). Since the data is not normally distributed, min-max scaling method would be better than the z-score method.

- Splitting:

xii. Data has been split into train and test sets as shown below in the example.

- X_train sample:

	age	economic_cond_national	economic_cond_household	Blair	Hague	Europe	political_knowledge	gender_male
529	70	4	3	2	4	10	2	0
141	62	4	3	5	2	1	2	0
1097	54	3	3	4	2	1	2	0
1010	76	4	4	5	2	11	0	0
663	37	4	3	4	4	10	2	1

- X_test sample:

	age	economic_cond_national	economic_cond_household	Blair	Hague	Europe	political_knowledge	gender_male
726	43	2	2	4	2	9	0	0
814	48	3	3	4	1	2	3	0
1474	56	4	3	2	4	8	0	1
1477	50	4	2	2	1	3	2	1
51	33	2	3	4	4	9	0	1

- y_train sample:

```
529     0
141     0
1097    0
1010    0
663     0
Name: vote, dtype: int64
```

- y_test sample:

```
726     0
814     0
1474    0
1477    0
51      0
Name: vote, dtype: int64
```

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

i. Logistic Regression model results:

On training data:

Accuracy = 83.41%

```
0.8341187558906692
[[657  69]
 [107 228]]
              precision    recall  f1-score   support

           0       0.86      0.90      0.88       726
           1       0.77      0.68      0.72       335

    accuracy                           0.83      1061
   macro avg       0.81      0.79      0.80      1061
weighted avg       0.83      0.83      0.83      1061
```

On test data:

Accuracy = 83.11%

```
0.831140350877193
[[306  25]
 [ 52  73]]
              precision    recall  f1-score   support

           0       0.85      0.92      0.89       331
           1       0.74      0.58      0.65       125

    accuracy                           0.83       456
   macro avg       0.80      0.75      0.77       456
weighted avg       0.82      0.83      0.82       456
```

xiii. LDA model results:

On train data:

Accuracy = 83.31%

```
0.8331762488218661
[[648  78]
 [ 99 236]]
              precision    recall  f1-score   support

           0       0.87      0.89      0.88       726
           1       0.75      0.70      0.73       335

    accuracy                           0.83      1061
   macro avg       0.81      0.80      0.80      1061
weighted avg       0.83      0.83      0.83      1061
```

On test data:

Accuracy = 83.11%

```
0.831140350877193
[[301  30]
 [ 47  78]]
              precision    recall  f1-score   support

           0       0.86      0.91      0.89       331
           1       0.72      0.62      0.67       125

    accuracy                           0.83       456
   macro avg       0.79      0.77      0.78       456
weighted avg       0.83      0.83      0.83       456
```

- **Observations on model validation:**

i. For both, Logit and LDA models, the accuracy score on train set is slightly higher than for the test set. Moreover, the accuracy score is **not** extraordinarily high on the training set followed by poor performance on the test set. With this, I concluded that the models are not overfit or underfit.

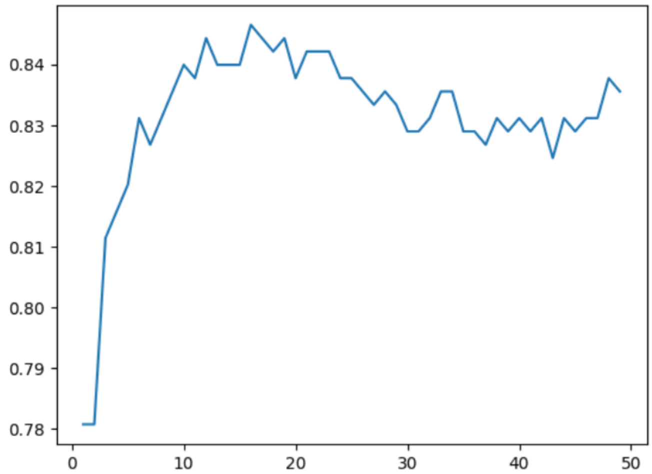
- **Inference for Logit and LDA:**

i. Both the models have performed decently on the test set with good precision score for class 1.
ii. The confusion matrix for both the models on test data shows us that the model was able to predict the class correctly for 379 test set records. It gave 77 wrong predictions. Note that these numbers are the same for both the models.
iii. In our prediction problem, both the classes are equally important and of interest. This gives us the freedom to use a model that gives a good score for either of the classes.
iv. We can see that the f1 scores for 0 class are relatively powerful.

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

    i. The following is a graph of model accuracy (y-axis) against k-values (x-axis) to get an idea of what k-value would be optimum when building the KNN model.

Figure 12 - n_neighbors vs. accuracy



    ii. KNN model results with parameters; n_neighbors=19 and weights='distance'

On training data:

Accuracy = 100%

```
1.0
[[726   0]
 [  0 335]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       726
           1       1.00      1.00      1.00       335

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

On test data:

Accuracy = 83.11%

```
0.831140350877193
[[304  27]
 [ 44  81]]
              precision    recall  f1-score   support

           0       0.87      0.92      0.90       331
           1       0.75      0.65      0.70       125

    accuracy                           0.84       456
   macro avg       0.81      0.78      0.80       456
weighted avg       0.84      0.84      0.84       456
```

iii. Naïve Bayes' model results:

On training data:

Accuracy = 82.94%

```
0.8294062205466541
              precision    recall  f1-score   support

           0       0.86      0.89      0.88       726
           1       0.75      0.70      0.72       335

    accuracy                           0.83      1061
   macro avg       0.81      0.79      0.80      1061
weighted avg       0.83      0.83      0.83      1061

[[646  80]
 [101 234]]
```

On test data:

Accuracy = 84.43%

```
0.8442982456140351
              precision    recall  f1-score   support

           0       0.88      0.90      0.89       331
           1       0.73      0.69      0.71       125

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

[[299  32]
 [ 39  86]]
```

iv. Model accuracy for Naïve Bayes' seems to have improved on testing data, which is rare.

- **Observations on validity of the models:**

i. KNN gave 1.00 accuracy on the training set. Moreover, it also gave an accuracy score of 0.83 on test set. The model is overfitting and not generalizing well on unseen data.

ii. A major concern is imbalanced data. However, after using SMOTE to balance the data, the model was still an overfit. Hence, I have reverted to the base model as the final one. At a later stage, I will optimize this model through Grid Search CV and tune it.

iii. Naïve Bayes' is not an overfit or underfit since it is performing good even on unseen data.

- **Inferences:**

i. For Naive Bayes' model, the base model is performing the best in comparison with SMOTE balanced training and test data sets.

ii. The model accuracy score on test data is slightly outperforming the model on training data.

iii. Score for train data: 82.94%. Score for test data: 84.43%.

iv. The model is performing better for the 0 class but since, according to our problem statement, both the parties are of interest, we can use the model to predict outcomes.

6. Model Tuning (4 pts), Bagging (1.5 pts) and Boosting (1.5 pts). Apply grid search on each model (include all models) and make models on best_params. Compare and comment on performances of all. Comment on feature importance if applicable. Successful implementation of both algorithms along with inferences and comments on the model performances.

Bagging:

i. Using Random Forest for bagging. Following are the results on training and test sets:

On training data:

```
0.998114985862394
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       726
           1       1.00      0.99      1.00       335

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

[[726   0]
 [  2 333]]
```

On test set:

```
0.8223684210526315
              precision    recall  f1-score   support

           0       0.86      0.90      0.88       331
           1       0.70      0.62      0.66       125

    accuracy                           0.82       456
   macro avg       0.78      0.76      0.77       456
weighted avg       0.82      0.82      0.82       456

[[297  34]
 [ 47  78]]
```

- Inferences for Random Forest (bagging):

i. The model has given 375 correct predictions and 81 false predictions for test data.
ii. The accuracy score on the training set is extremely high, i.e., 99.81% but falls steeply for the test set, for which the model accuracy is 82.23%.
iii. This suggests that the model is overfitting the data and capturing too much noise. It will perform badly on unseen data and can't be trusted.

Boosting:

i. AdaBoosting model results:

On train set:

```
0.8416588124410933
              precision    recall  f1-score   support

           0       0.87      0.90      0.89       726
           1       0.77      0.71      0.74       335

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

[[654  72]
 [ 96 239]]
```

On test set:

```
0.8289473684210527
              precision    recall  f1-score   support

           0       0.87      0.90      0.88       331
           1       0.70      0.65      0.68       125

    accuracy                           0.83       456
   macro avg       0.79      0.77      0.78       456
weighted avg       0.83      0.83      0.83       456

[[297  34]
 [ 44  81]]
```

- Inferences for AdaBoost:

i. F-1 score is good for both training and test datasets. Moreover, the accuracy score of 84.17% on training data has not dropped significantly on the testing data, for which the accuracy is 82.89%.

ii. We can rely on the model for predicting class 0 as the precision and recall metrics are also high.

iii. It has given 378 correct and 78 false predictions on the test set.

ii. Gradient Boosting results:

On train set:

```
0.8680490103675778
              precision    recall  f1-score   support

           0       0.89      0.93      0.91       726
           1       0.82      0.74      0.78       335

    accuracy                           0.87      1061
   macro avg       0.85      0.83      0.84      1061
weighted avg       0.87      0.87      0.87      1061

[[673  53]
 [ 87 248]]
```

On test set:

```
0.8442982456140351
              precision    recall  f1-score   support

           0       0.88      0.92      0.90       331
           1       0.75      0.66      0.70       125

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

[[303  28]
 [ 43  82]]
```

- Inferences for Gradient Boost:

i. F-1 score is good for both training and test datasets with no significant change. Moreover, the accuracy score of 86.8% on training data has not dropped significantly on the testing data, for which the accuracy is 84.43%.

ii. Here too, we can rely on the model for predicting class 0 as the precision and recall metrics are also high.

iii. It has given 385 correct and 71 false predictions on the test set.

Till now, between all bagging and boosting models, Gradient Boost seems to perform the best with highest correct predictions, accuracy and f-1 score.

Model Tuning with Grid Search CV:

i. **Logit:** Best parameters given are as follows – {'penalty': 'none', 'solver': 'sag', 'tol': 0.0001}

Classification report along with accuracy score and confusion matrix given below:

Train

```
0.8360037700282752
[[655  71]
 [103 232]]
              precision    recall  f1-score   support

           0       0.86      0.90      0.88       726
           1       0.77      0.69      0.73       335

    accuracy                           0.84      1061
   macro avg       0.81      0.80      0.81      1061
weighted avg       0.83      0.84      0.83      1061
```

Test

```
0.831140350877193
[[304  27]
 [ 50  75]]
              precision    recall  f1-score   support

           0       0.86      0.92      0.89       331
           1       0.74      0.60      0.66       125

    accuracy                           0.83       456
   macro avg       0.80      0.76      0.77       456
weighted avg       0.82      0.83      0.83       456
```

a) For Logit, the base model is still slightly outperforming the model on which Grid Search CV was applied. Due to this, I will choose to use the base model with an accuracy score of 0.8333 instead of the grid search model with the score of 0.8311

ii. **LDA:** For LDA, the default parameter of _solver='svd' is performing the best. Therefore, the base model is already tuned for optimal performance.

iii. **KNN:** Best parameters given are as follows – {'leaf_size': 1, 'n_neighbors': 33, 'weights': 'uniform'}

Classification report along with accuracy score and confusion matrix given below:

Train

```
0.8303487276154571
[[660  66]
 [114 221]]
              precision    recall  f1-score   support

           0       0.85      0.91      0.88       726
           1       0.77      0.66      0.71       335

    accuracy                           0.83      1061
   macro avg       0.81      0.78      0.80      1061
weighted avg       0.83      0.83      0.83      1061
```

Test

```
0.831140350877193
[[304  27]
 [ 50  75]]
              precision    recall  f1-score   support

           0       0.86      0.92      0.89       331
           1       0.74      0.60      0.66       125

    accuracy                           0.83       456
   macro avg       0.80      0.76      0.77       456
weighted avg       0.82      0.83      0.83       456
```

a) For KNN, Grid Search CV has given different hyperparameters for optimized performance. While the base model had 'weights='distance'', the tuned model has weights set to 'uniform'. The n_neighbors parameter has also been changed from 19 to 33 and is solving the problem of overfitting, hence, making the model stable. I'm finalizing this model for use on unseen data.

b) Additionally, the KNN model is working well with leaf_size=1.

iv. **Naïve Bayes':** Best parameter given are as follows – {'var_smoothing': 1e-323}

Classification report along with accuracy score and confusion matrix given below:

Train

```
0.8294062205466541
              precision    recall  f1-score   support

           0       0.86      0.89      0.88       726
           1       0.75      0.70      0.72       335

    accuracy                           0.83      1061
   macro avg       0.81      0.79      0.80      1061
weighted avg       0.83      0.83      0.83      1061

[[646  80]
 [101 234]]
```

Test

```
0.8442982456140351
              precision    recall  f1-score   support

           0       0.88      0.90      0.89       331
           1       0.73      0.69      0.71       125

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

[[299  32]
 [ 39  86]]
```

v. **Random Forest:** Best parameter given are as follows – {'bootstrap': False, 'max_depth': 3, 'max_features': 5, 'min_samples_leaf': 20, 'min_samples_split': 30, 'n_estimators': 40}

Classification report along with accuracy score and confusion matrix given below:

Train

```
0.9981149858623940
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       726
           1       1.00      0.99      1.00       335

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

[[726   0]
 [  2 333]]
```

Test

```
0.8223684210526315
              precision    recall  f1-score   support

           0       0.86      0.90      0.88       331
           1       0.70      0.62      0.66       125

    accuracy                           0.82       456
   macro avg       0.78      0.76      0.77       456
weighted avg       0.82      0.82      0.82       456

[[304  27]
 [ 44  81]]
```

vi. **Adaptive Boosting:** Best parameter given are as follows – {'algorithm': 'SAMME', 'learning_rate': 1.0, 'n_estimators': 50}

Classification report along with accuracy score and confusion matrix given below:

Train	Test

0.8407163053722903

	precision	recall	f1-score	support
0	0.87	0.90	0.89	726
1	0.77	0.71	0.74	335
accuracy			0.84	1061
macro avg	0.82	0.81	0.81	1061
weighted avg	0.84	0.84	0.84	1061

```
[[654  72]
 [ 97 238]]
```

0.8377192982456141

	precision	recall	f1-score	support
0	0.88	0.90	0.89	331
1	0.72	0.66	0.69	125
accuracy			0.84	456
macro avg	0.80	0.78	0.79	456
weighted avg	0.83	0.84	0.84	456

```
[[299  32]
 [ 42  83]]
```

a) Tuning AdaBoost is showing slight performance boost in both 'accuracy' as well as f1-score. Therefore, keeping abcl2 as the final model for AdaBoost.

vii. **Gradient Boosting:** Best parameter given are as follows – {'learning_rate': 0.1, 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}

Classification report along with accuracy score and confusion matrix given below:

Train

0.8868991517436381

	precision	recall	f1-score	support
0	0.90	0.94	0.92	726
1	0.85	0.78	0.81	335
accuracy			0.89	1061
macro avg	0.88	0.86	0.87	1061
weighted avg	0.89	0.89	0.89	1061

```
[[680  46]
 [ 74 261]]
```

Test

0.8399122807017544

	precision	recall	f1-score	support
0	0.88	0.91	0.89	331
1	0.73	0.66	0.69	125
accuracy			0.84	456
macro avg	0.80	0.79	0.79	456
weighted avg	0.84	0.84	0.84	456

```
[[300  31]
 [ 42  83]]
```

a) The GradientBoost base model is still performing better in terms of f1-score and model accuracy for test data. Hence, will keep the base as final model.

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

- AUC score, ROC curve and confusion matrix for Final Models:

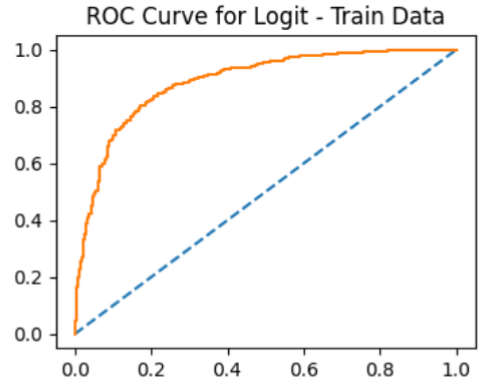
i. Logit:

a) Train AUC Score = 0.887

ROC Curve:

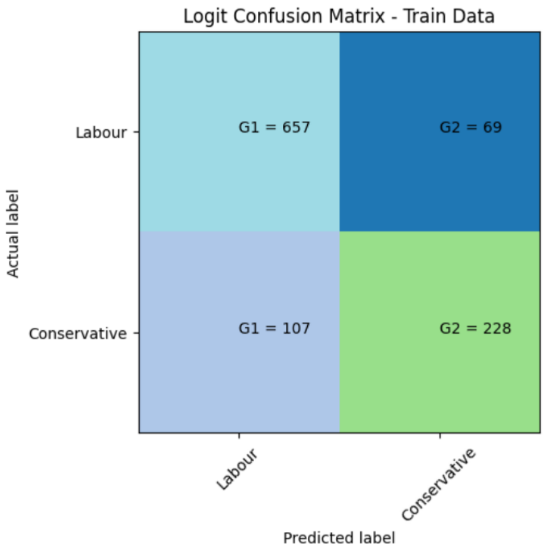
Figure 13 - Logit AUC_ROC Curve I

AUC: 0.887



Confusion matrix:

Figure 14 - Logit Confusion Matrix I



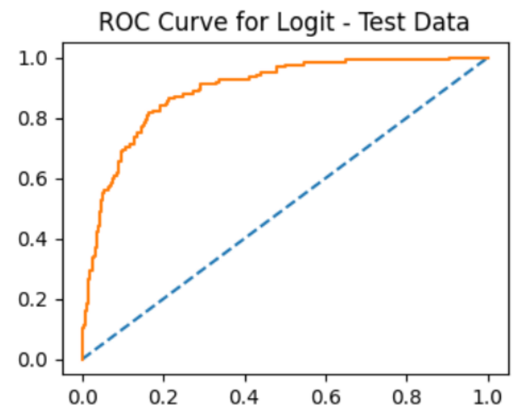
Classification report: (accuracy=0.8341)

```
0.8341187558906692
[[657  69]
 [107 228]]
              precision    recall  f1-score   support

           0       0.86      0.90      0.88       726
           1       0.77      0.68      0.72       335

    accuracy                           0.83      1061
   macro avg       0.81      0.79      0.80      1061
weighted avg       0.83      0.83      0.83      1061
```

b) Test AUC Score = 0.892

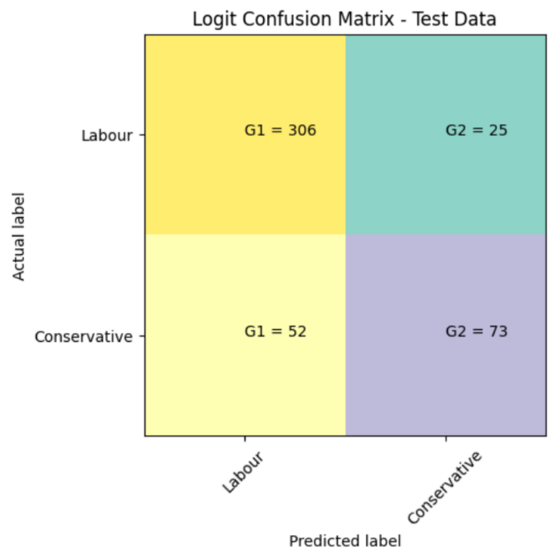
Figure 15 - Logit AUC_ROC Curve II

```
AUC: 0.892
```



Confusion matrix:

Figure 16 - Logit Confusion Matrix II



Classification report: (accuracy=0.8311)

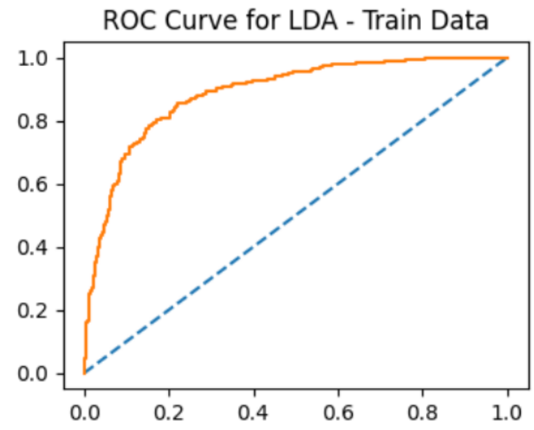
```
0.831140350877193
[[306  25]
 [ 52  73]]
              precision    recall  f1-score   support

           0       0.85      0.92      0.89       331
           1       0.74      0.58      0.65       125

    accuracy                           0.83       456
   macro avg       0.80      0.75      0.77       456
weighted avg       0.82      0.83      0.82       456
```

ii. LDA:

a) Train AUC Score = 0.887

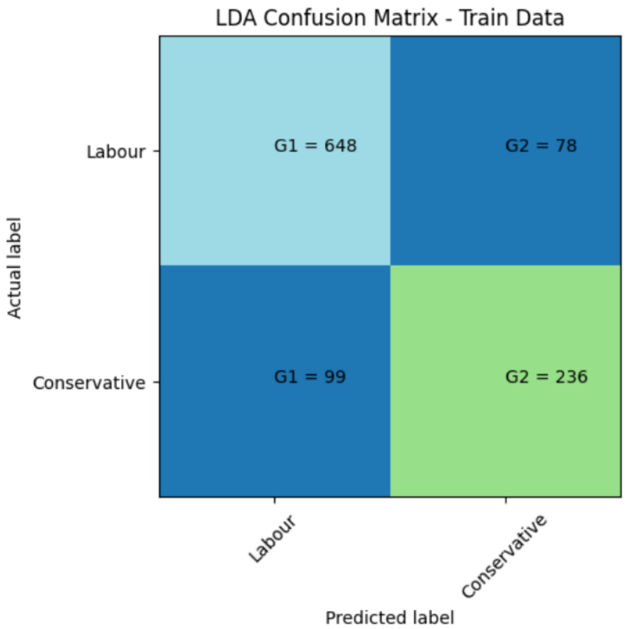
Figure 17 - LDA AUC_ROC Curve I

```
AUC: 0.887
```



Confusion matrix:

Figure 18 - LDA Confusion Matrix I

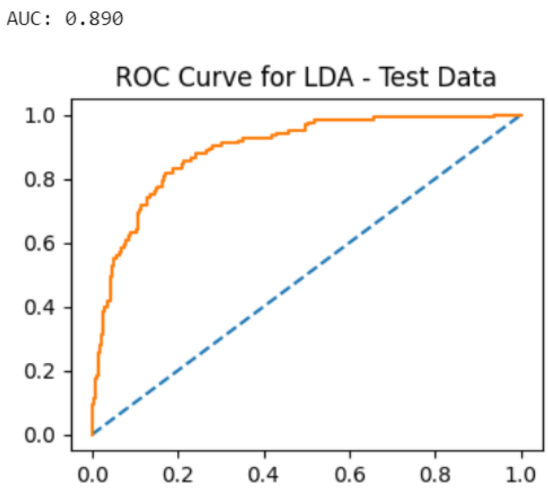


Classification report: (accuracy=0.8331)

```
0.8331762488218661
[[648  78]
 [ 99 236]]
              precision    recall  f1-score   support

           0       0.87      0.89      0.88       726
           1       0.75      0.70      0.73       335

    accuracy                           0.83      1061
   macro avg       0.81      0.80      0.80      1061
weighted avg       0.83      0.83      0.83      1061
```

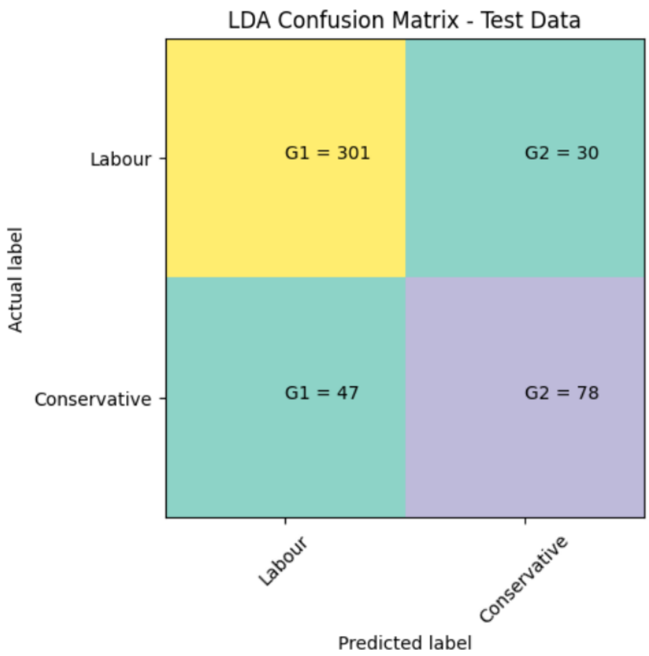
b) Test AUC Score = 0.890

Figure 19 - LDA AUC_ROC Curve II



Confusion matrix:

Figure 20 - LDA Confusion Matrix II



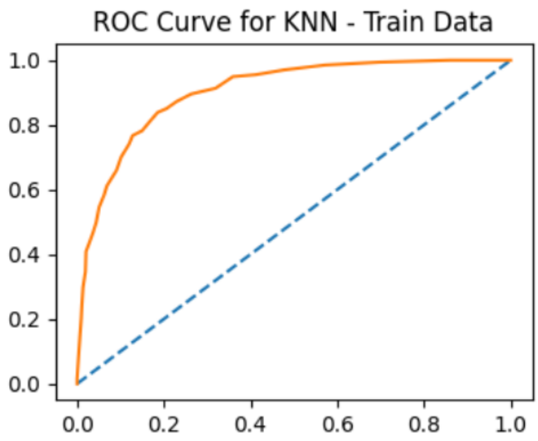
Classification report: (accuracy=0.8311)

```
0.831140350877193
[[301  30]
 [ 47  78]]
              precision    recall  f1-score   support

           0       0.86      0.91      0.89       331
           1       0.72      0.62      0.67       125

    accuracy                           0.83       456
   macro avg       0.79      0.77      0.78       456
weighted avg       0.83      0.83      0.83       456
```

iii. KNN:

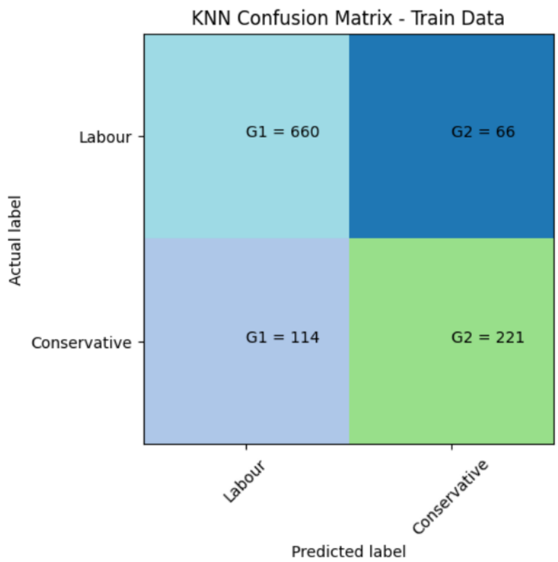
a) Train AUC Score = 0.902

Figure 21 - KNN AUC_ROC Curve I



Confusion matrix:

Figure 22 - KNN Confusion Matrix I



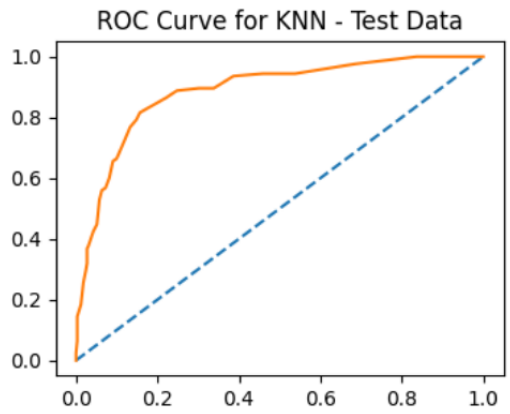
Classification report: (accuracy=0.8303)

```
0.8303487276154571
[[660  66]
 [114 221]]
              precision    recall  f1-score   support

           0       0.85      0.91      0.88       726
           1       0.77      0.66      0.71       335

    accuracy                           0.83      1061
   macro avg       0.81      0.78      0.80      1061
weighted avg       0.83      0.83      0.83      1061
```

b) Test AUC Score = 0.886

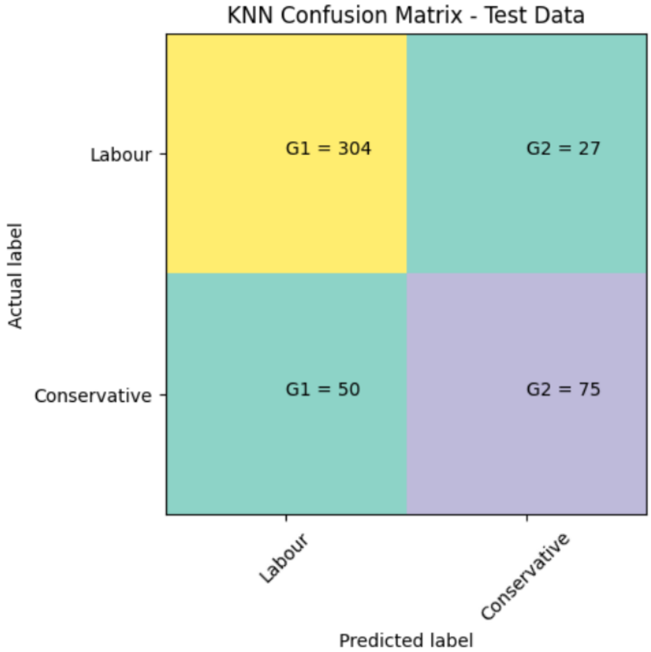
Figure 23 - KNN AUC_ROC Curve II

AUC: 0.886



Confusion matrix:

Figure 24 - KNN Confusion Matrix II



Classification report: (accuracy=0.8311)

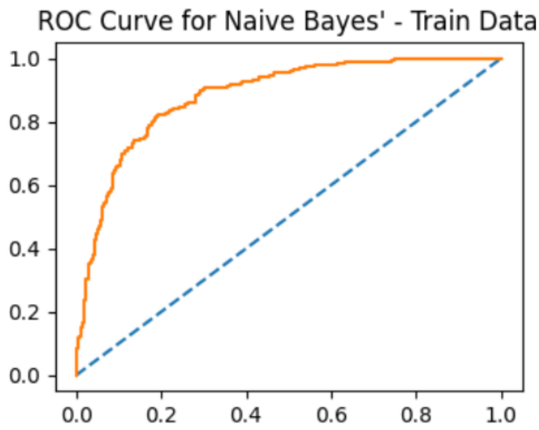
```
0.831140350877193
[[304  27]
 [ 50  75]]
              precision    recall  f1-score   support

           0       0.86      0.92      0.89       331
           1       0.74      0.60      0.66       125

    accuracy                           0.83       456
   macro avg       0.80      0.76      0.77       456
weighted avg       0.82      0.83      0.83       456
```

iv. Naïve Bayes':

a) Train AUC Score = 0.883

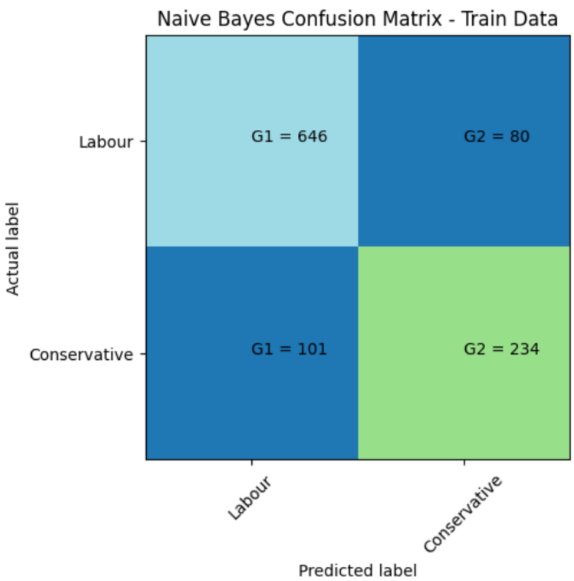
Figure 25 - Naive Bayes' AUC_ROC Curve I

```
AUC: 0.883
```



Confusion matrix:

Figure 26 - Naive Bayes' Confusion Matrix I



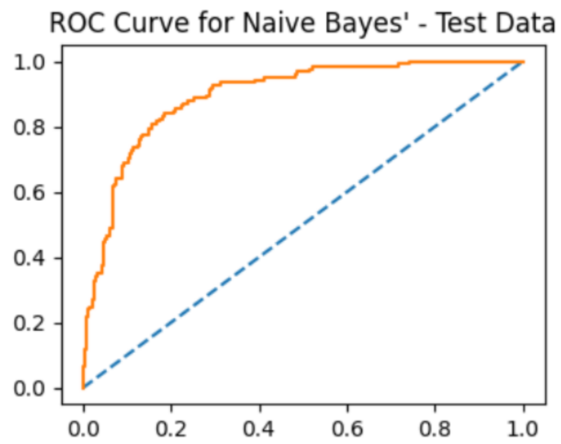
Classification report: (accuracy=0.8294)

```
0.8294062205466541
              precision    recall  f1-score   support

           0       0.86      0.89      0.88       726
           1       0.75      0.70      0.72       335

    accuracy                           0.83      1061
   macro avg       0.81      0.79      0.80      1061
weighted avg       0.83      0.83      0.83      1061

[[646  80]
 [101 234]]
```

b) Test AUC Score = 0.895

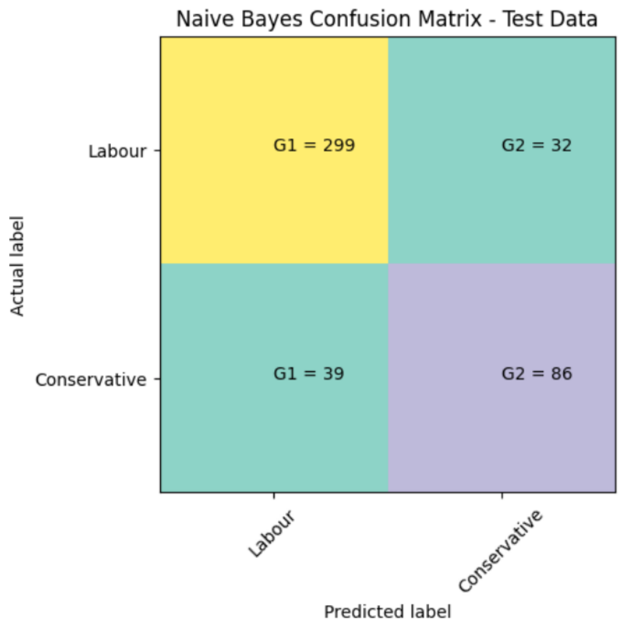
Figure 27 - Naive Bayes' AUC_ROC Curve II

AUC: 0.895



Confusion matrix:

Figure 28 - Naive Bayes' Confusion Matrix II



Classification report: (accuracy=0.8443)

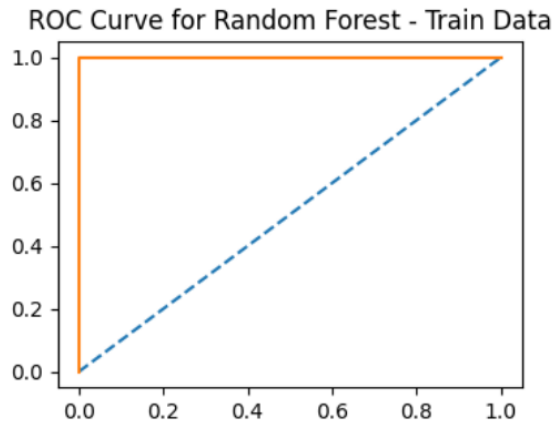
```
0.8442982456140351
              precision    recall  f1-score   support

           0       0.88      0.90      0.89       331
           1       0.73      0.69      0.71       125

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

[[299  32]
 [ 39  86]]
```

v. Random Forest:

a) Train AUC Score = 1.000

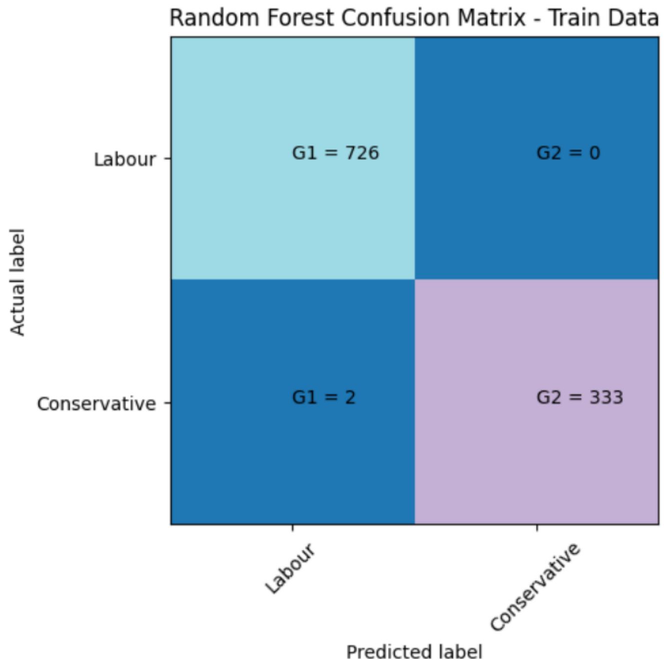
Figure 29 - Random Forest AUC_ROC Curve I

AUC: 1.000



Confusion matrix:

Figure 30 - Random Forest Confusion Matrix I



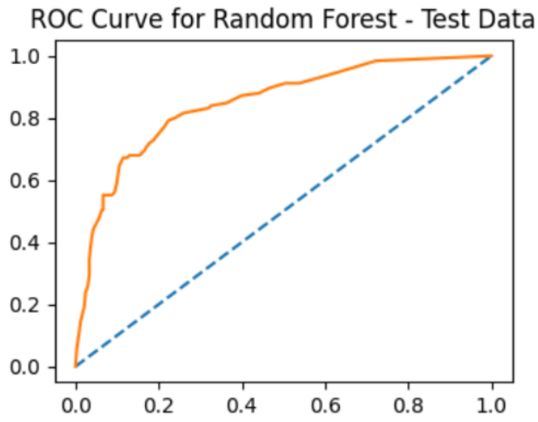
Classification report: (accuracy=0.9981)

```
0.998114985862394
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       726
           1       1.00      0.99      1.00       335

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

[[726   0]
 [  2 333]]
```

b) Test AUC Score = 0.847

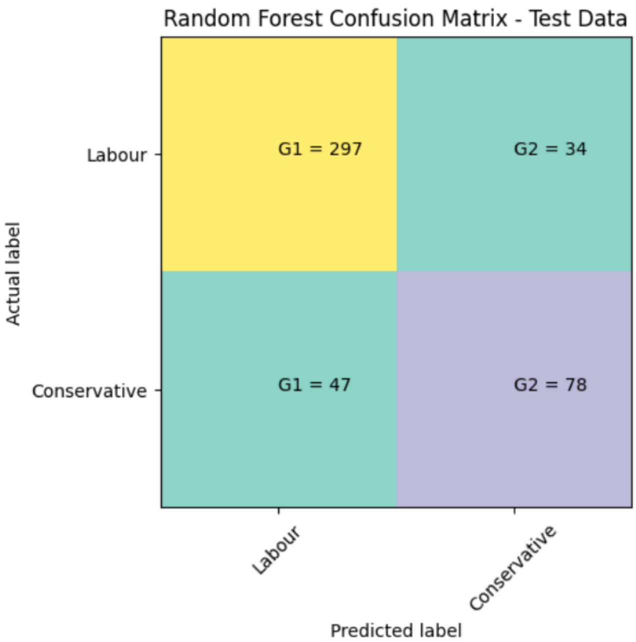
Figure 31 - Random Forest AUC_ROC Curve II

```
AUC: 0.847
```



Confusion matrix:

Figure 32 - Random Forest Confusion Matrix II



Classification report: (accuracy=0.8224)

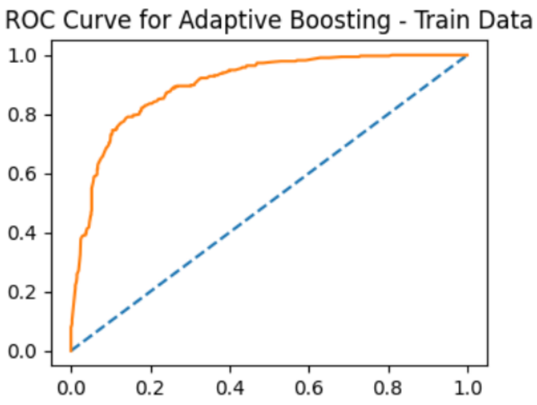
```
0.8223684210526315
              precision    recall  f1-score   support

           0       0.86      0.90      0.88       331
           1       0.70      0.62      0.66       125

    accuracy                           0.82       456
   macro avg       0.78      0.76      0.77       456
weighted avg       0.82      0.82      0.82       456

[[297  34]
 [ 47  78]]
```

vi. Adaptive Boosting:

a) Train AUC Score = 0.899

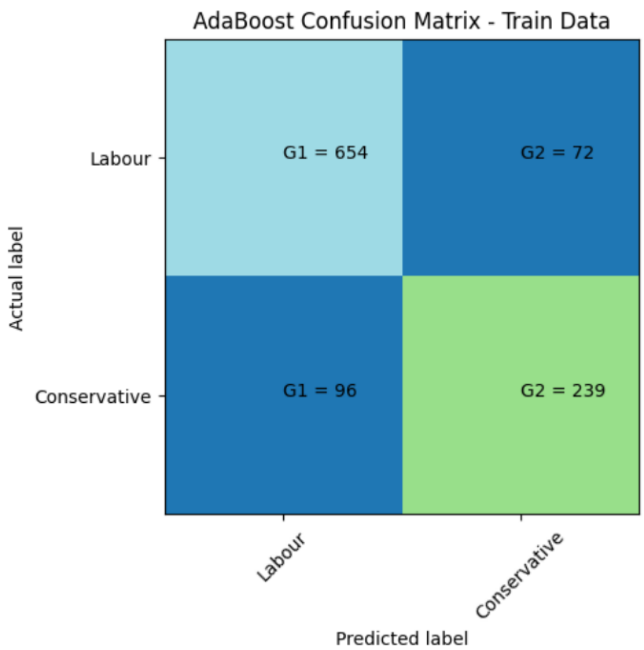
Figure 33 - AdaBoost AUC_ROC Curve I

AUC: 0.899



Confusion matrix:

Figure 34 - AdaBoost Confusion Matrix I



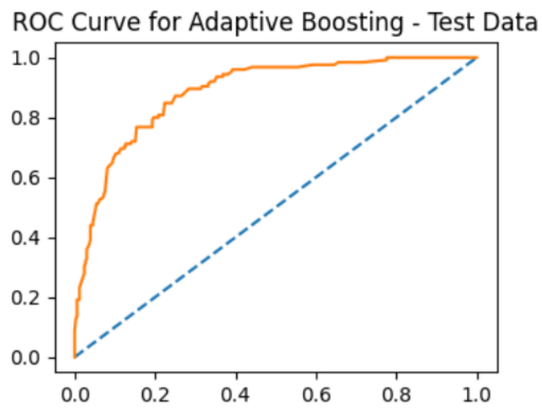
Classification report: (0.8417)

```
0.8416588124410933
              precision    recall  f1-score   support

           0       0.87      0.90      0.89       726
           1       0.77      0.71      0.74       335

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

[[654  72]
 [ 96 239]]
```

b) Test AUC Score = 0.886

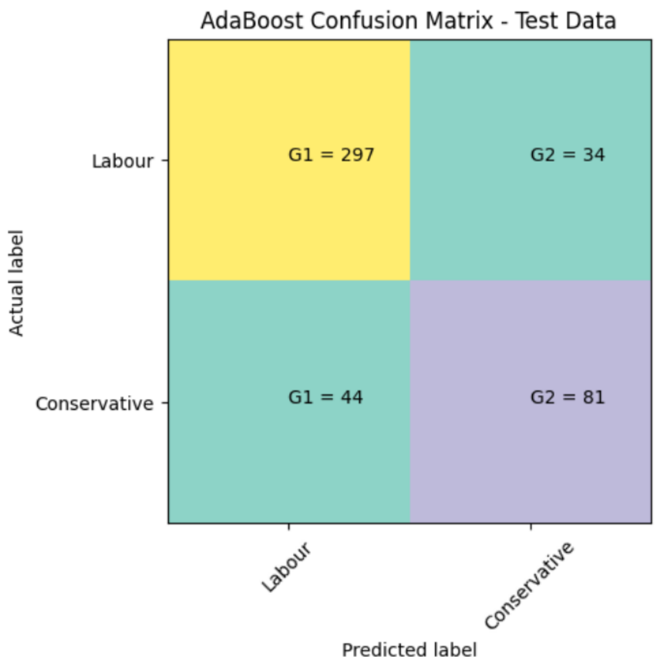
Figure 35 - AdaBoost AUC_ROC Curve II

```
AUC: 0.886
```



Confusion matrix:

Figure 36 - AdaBoost Confusion Matrix II



Classification report:

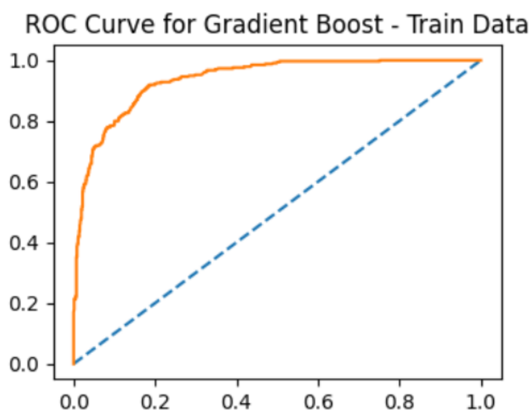
```
0.8289473684210527
              precision    recall  f1-score   support

           0       0.87      0.90      0.88       331
           1       0.70      0.65      0.68       125

    accuracy                           0.83       456
   macro avg       0.79      0.77      0.78       456
weighted avg       0.83      0.83      0.83       456

[[297  34]
 [ 44  81]]
```

vii. Gradient Boosting:

a) Train AUC Score = 0.938

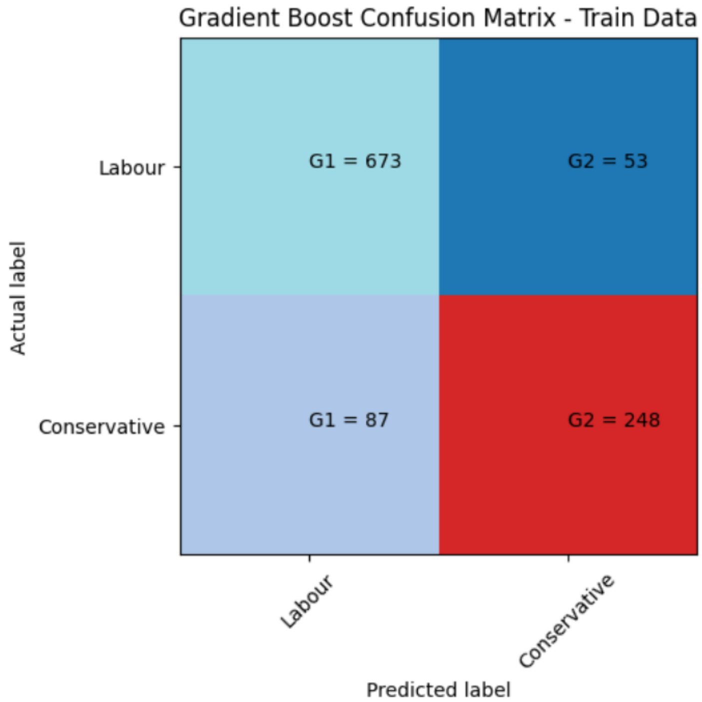
Figure 37 - Gradient Boost AUC_ROC Curve I

```
AUC: 0.938
```



Confusion matrix:

Figure 38 - Gradient Boost Confusion Matrix I



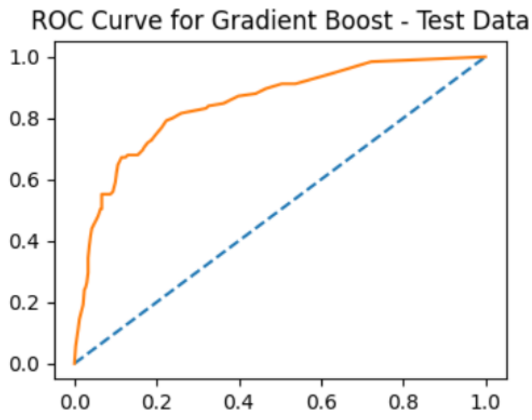
Classification report: (accuracy=0.8680)

```
0.8680490103675778
              precision    recall  f1-score   support

           0       0.89      0.93      0.91       726
           1       0.82      0.74      0.78       335

    accuracy                           0.87      1061
   macro avg       0.85      0.83      0.84      1061
weighted avg       0.87      0.87      0.87      1061

[[673  53]
 [ 87 248]]
```

b) Test AUC Score = 0.847

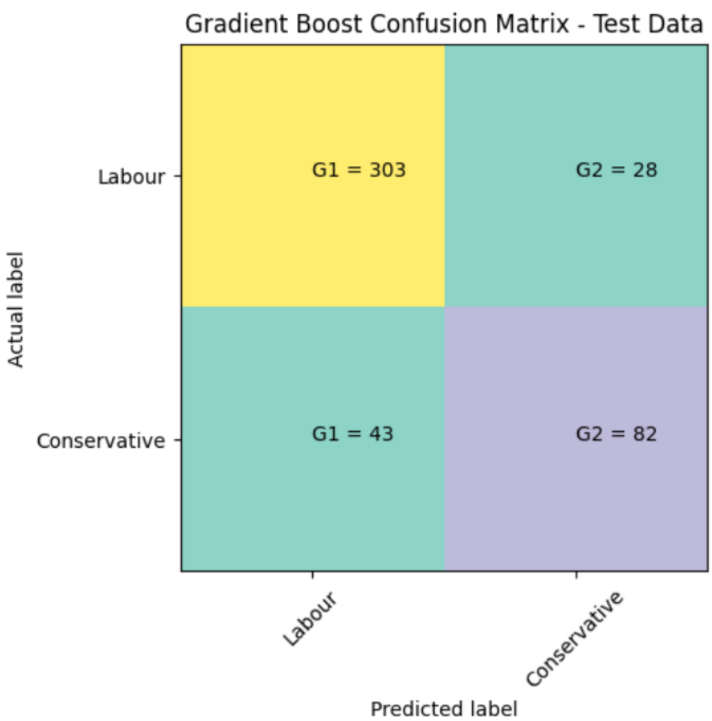
Figure 39 - Gradient Boost AUC_ROC Curve

AUC: 0.847



Confusion matrix:

Figure 40 - Gradient Boost Confusion Matrix II



Classification report: (accuracy=0.8443)

```
0.8442982456140351
              precision    recall  f1-score   support

           0       0.88      0.92      0.90       331
           1       0.75      0.66      0.70       125

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

[[303  28]
 [ 43  82]]
```

- **Comparison of all final models in tabular form:**

	model	accuracy	precision	recall	f1-score	AUC	remark
0	Logistic Regression	0.8311	0.85	0.92	0.89	0.892	None
1	LDA	0.8311	0.86	0.91	0.89	0.890	None
2	KNN	0.8312	0.86	0.92	0.89	0.886	None
3	Naive Bayes	0.8443	0.88	0.90	0.89	0.895	None
4	Random Forest	0.8224	0.86	0.90	0.88	0.847	Overfit
5	AdaBoost	0.8377	0.88	0.90	0.89	0.886	None
6	Gradient Boost	0.8443	0.88	0.92	0.90	0.847	Possible Overfit

- *Conclusions:*

i. I will go ahead with Logistic Regression or Naive Bayes' since the models seem to generalize well on unseen data. They are also giving good AUC scores.

ii. It is worth noting that Grid Search CV optimized the KNN model well when we tuned its hyperparameters. At first trial, the model was an overfit but after changing the 'weights', 'leaf_size', and 'n_neighbors', the model performed much better on unseen data.

iii. Feature importance through the built-in ‘feature_importances_’ attribute for models where it’s applicable:

For random forest –

```
                              Imp
age                      0.219386
economic_cond_national   0.076567
economic_cond_household  0.069362
Blair                    0.139384
Hague                    0.234336
Europe                   0.146279
political_knowledge      0.083348
gender_male              0.031338
```

For AdaBoost –

	Imp
age	0.105706
economic_cond_national	0.059043
economic_cond_household	0.000000
Blair	0.344797
Hague	0.155583
Europe	0.298007
political_knowledge	0.036864
gender_male	0.000000

For Gradient Boost –

	Imp
age	0.078621
economic_cond_national	0.054884
economic_cond_household	0.013909
Blair	0.214824
Hague	0.406475
Europe	0.113464
political_knowledge	0.117433
gender_male	0.000390

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

- Final models: **Logit and Naive Bayes'**
  i. According to the business problem, we can take any class as the class of interest since Labour and Conservative parties hold equal value in predicting votes. Due to the problem of class imbalance, the models are performing better for the 0 class (CLabour). Therefore, I have considered the f1-scores for 0 class to evaluate the models.
  ii. The final models are Logistic Regression (log_model) and Naive Bayes' (model_nb) since they have performed the best in terms of overall metrics, such as 'accuracy', 'f1-score', 'AUC_ROC score'.
  iii. According to the Logit model, the following is the equation that measures the dependence of Y variable on different features:

  Y = -2.45857996 + age(1.03420336) + economic_cond_national(-1.22634905) + economic_cond_household(-0.37046377) + Blair(-2.32365965) + Hague(3.18940252) + Europe(1.80540481) + political_knowledge(1.10605656) + gender_male(-0.03399354)

  - The most important features, according to the above Logit equation are:

    a. Hague, direct impact
    b. Blair, *inverse impact*
    c. Europe, *direct impact*

d. economic_cond_national, *inverse impact*

iv. According to Naïve Bayes', the following is a sample of the probabilities for each class:

	proba_0	proba_1
0	1.0	0.000000e+00
1	1.0	1.564534e-253
2	1.0	7.693498e-228
3	1.0	6.239515e-110
4	1.0	0.000000e+00
...	...	...
983	1.0	0.000000e+00
1154	1.0	0.000000e+00
1236	1.0	0.000000e+00
1244	1.0	0.000000e+00
1438	1.0	0.000000e+00

1525 rows × 2 columns

- In the above example, Naive Bayes' has predicted probabilities of each class (0 and 1) for all 1525 rows, including the test dataset (unseen data).

Problem 2

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

a. President Franklin D. Roosevelt in 1941
b. President John F. Kennedy in 1961
c. President Richard Nixon in 1973

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

- Speech 1 (Roosevelt):
  i. Total word count = 1344
  ii. Total character count = 5381
  iii. Total sentence count = 68

- Speech 2 (Kennedy):
  iv. Total word count = 1370
  v. Total character count = 5484
  vi. Total sentence count = 52

- Speech 3 (Nixon):
  vii. Total word count = 1816
  viii. Total character count = 7099
  ix. Total sentence count = 68

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

- Sample sentence (text) for Roosevelt after removing the stopwords:

  "nation day inaugur sinc peopl renew sens dedic unit state washington day task peopl creat weld togeth nation lincoln day task peopl preserv nation disrupt within day task peopl save nation institut disrupt without us come time midst swift happen paus moment take stock recal place histori rediscov may risk real peril inact live nation determin count year lifetim human spirit life man three-scor year ten littl littl less life nation full measur live men doubt men believ democraci form govern"

  Word count before: 1344
  Word count after: 630

- Sample sentence (text) for Kennedy after removing the stopwords:

  "vice presid johnson speaker chief justic presid eisenhow vice presid nixon presid truman reverend clergi fellow citizen observ today victori parti celebr freedom symbol end well begin signifi renew well chang sworn befor almighti god solemn oath forebear l prescrib near centuri three quarter ago world veri differ man hold mortal hand power abolish form human poverti form human life yet revolutionari belief forebear fought still issu around globe belief right man come generos state hand god"

  Word count before: 1370
  Word count after: 704

- Sample sentence (text) for Nixon after removing the stopwords:

  "vice presid speaker chief justic senat cook eisenhow fellow citizen great good countri share togeth met four year ago america bleak spirit depress prospect seem endless war abroad destruct conflict home meet today stand threshold new era peac world central question befor us shall use peac let us resolv era enter postwar period often time retreat isol lead stagnat home invit new danger abroad let us resolv becom time great respons great born renew spirit promis america enter third centuri nation"

  Word count before: 1816
  Word count after: 846

NOTE: Please consider that the sample text has been pre-processed, which includes stemming. That's the reason the text looks stemmed after removing the stopwords.

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

- *Top 3 frequent words for Roosevelt's speech*:

i. 'nation' = 17 times
ii. 'know' = 10 times
iii. 'peopl' = 9 times

- *Top 3 frequent words for Kennedy's speech*:

i. 'let' = 16 times
ii. 'us' = 12 times
iii. 'power' = 9 times

- *Top 3 frequent words for Nixon's speech*:

i. 'us' = 26 times
ii. 'let' = 22 times
iii. 'america' = 21 times

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

- Word cloud for Roosevelt's speech:

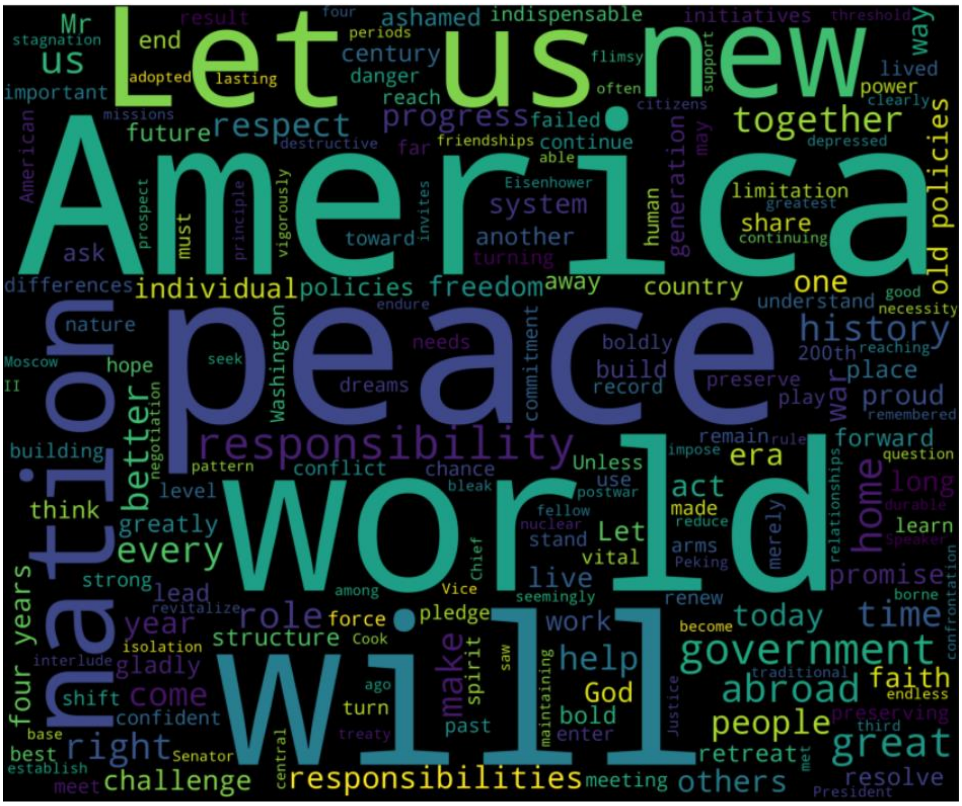
Figure 41 - Word Cloud I

- Word cloud for Kennedy’s speech:

Figure 42 - Word Cloud II

- Word cloud for Nixon’s speech:

Figure 43 - Word Cloud III



-----THE DOCUMENT ENDS HERE-----