MACHINE LEARNING-2 PROJECT REPORT

By Suraj Mishra

Problem 1

Context

CNBE, a prominent news channel, is gearing up to provide insightful coverage of recent elections, recognizing the importance of data-driven analysis. A comprehensive survey has been conducted, capturing the perspectives of 1525 voters across various demographic and socio-economic factors. This dataset encompasses 9 variables, offering a rich source of information regarding voters' characteristics and preferences.

Objective

The primary objective is to leverage machine learning to build a predictive model capable of forecasting which political party a voter is likely to support. This predictive model, developed based on the provided information, will serve as the foundation for creating an exit poll. The exit poll aims to contribute to the accurate prediction of the overall election outcomes, including determining which party is likely to secure the majority of seats.

Problem 2

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

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

Define the problem and perform Exploratory Data Analysis - Problem definition - Check shape, Data types, and statistical summary - Univariate analysis - Multivariate analysis - Use appropriate visualizations to identify the patterns and insights - Key meaningful observations on individual variables and the relationship between variables	6
Data Pre-processing Prepare the data for modelling: - Outlier Detection(treat, if needed)) - Encode the data - Data split - Scale the data (and state your reasons for scaling the features)	

Model Building
- Metrics of Choice (Justify the evaluation metrics) - Model Building (KNN, Naive bayes, Bagging, Boosting)

Model Performance evaluation
- Check the confusion matrix and classification metrics for all the models (for both train and test dataset) - ROC-AUC score and plot the curve - Comment on all the model performance

Model Performance improvement
- Improve the model performance of bagging and boosting models by tuning the model - Comment on the model performance improvement on training and test data

Final Model Selection
- Compare all the model built so far - Select the final model with the proper justification - Check the most important features in the final model and draw inferences.

Actionable Insights & Recommendations
- Compare all four models - Conclude with the key takeaways for the business

Problem 2 - Define the problem and Perform Exploratory Data Analysis
-Problem Definition - Find the number of Character, words & sentences in all three speeches

Problem 2 - Text cleaning
- Stop-word removal - Stemming - find the 3 most common words used in all three speeches

Problem 2 - Plot Word cloud of all three speeches
- Show the most common words used in all three speeches in the form of word clouds

---

Define the problem and perform Exploratory Data Analysis
- Problem definition - Check shape, Data types, and statistical summary - Univariate analysis - Multivariate analysis - Use appropriate visualizations to identify the patterns and insights - Key meaningful observations on individual variables and the relationship between variables

Data Information:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1525 entries, 1 to 1525
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: 119.1+ KB
```

Observation:

• The 'unnamed' column from the dataset was dropped as it was not useful for our study.
• The dataset initially had 8 duplicated values, which were subsequently dropped.

• After dropping the duplicate values, the dataset contained 1517 rows and 9 columns. Initially, it had 1525 rows and 9 columns.
• The dataset consists of 7 numerical data types and 2 categorical data types.
• There are no null values present in any column of the dataset.

Checking for missing values:

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

Checking the skewness of the data:

```
vote                       0.858449
age                        0.144621
economic.cond.national    -0.240453
economic.cond.household   -0.149552
Blair                     -0.535419
Hague                      0.152100
Europe                    -0.135947
political.knowledge       -0.426838
gender                     0.130239
dtype: float64
```

• The rule of thumb for skewness indicates that:

- Skewness between -0.5 and 0.5 suggests symmetrical data.
- Skewness between -1 and -0.5, or between 0.5 and 1, suggests moderately skewed data.
- Skewness less than -1 or greater than 1 suggests highly skewed data.

• Based on the analysis, the data in your dataset do not exhibit significant skewness. The skewness values are all within the range of -0.5 to 0.5, indicating symmetrical data.

• It’s noted that the skewness value for the 'Blair' variable is slightly higher than -0.5, but still within the symmetrical range.

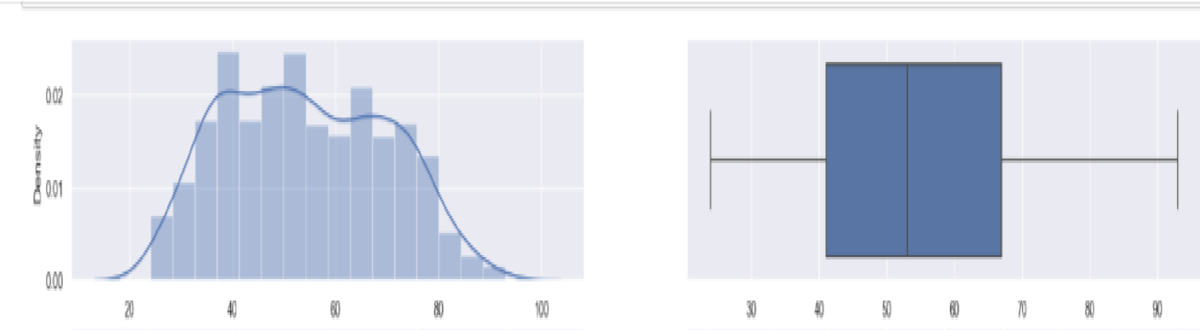
• Overall, the dataset shows a symmetrical distribution of values based on the skewness analysis.

Univariate Analysis:

Description of 'age':

	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

Histogram and Box plot of age

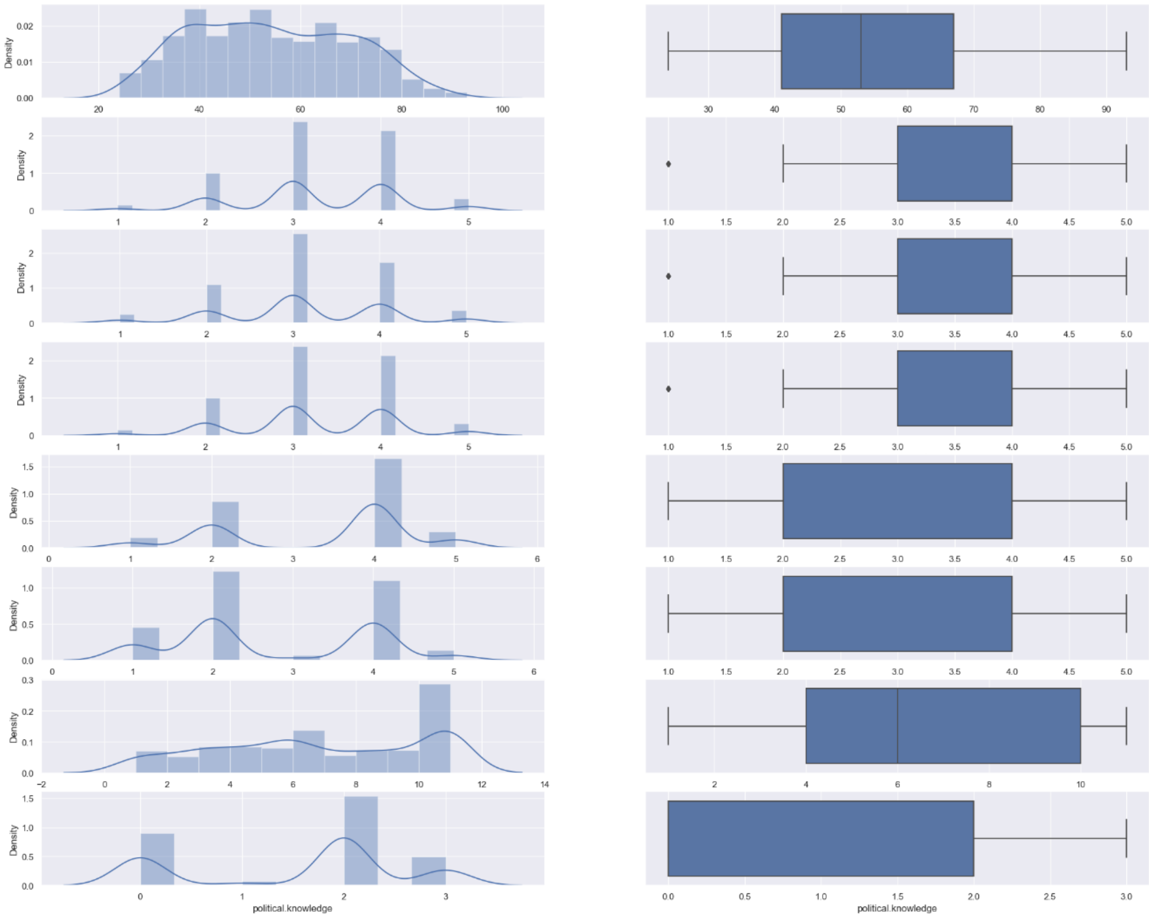


Observation

• The data follows a normal distribution.
• The majority of individuals fall within the age range of 40 to 70 years.
• No outliers were detected in the dataset.

• The age values range from a minimum of 24 years to a maximum of 93 years.
• The mean age of the dataset is 54.241266 years.

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



Observations on Numerical Variables:

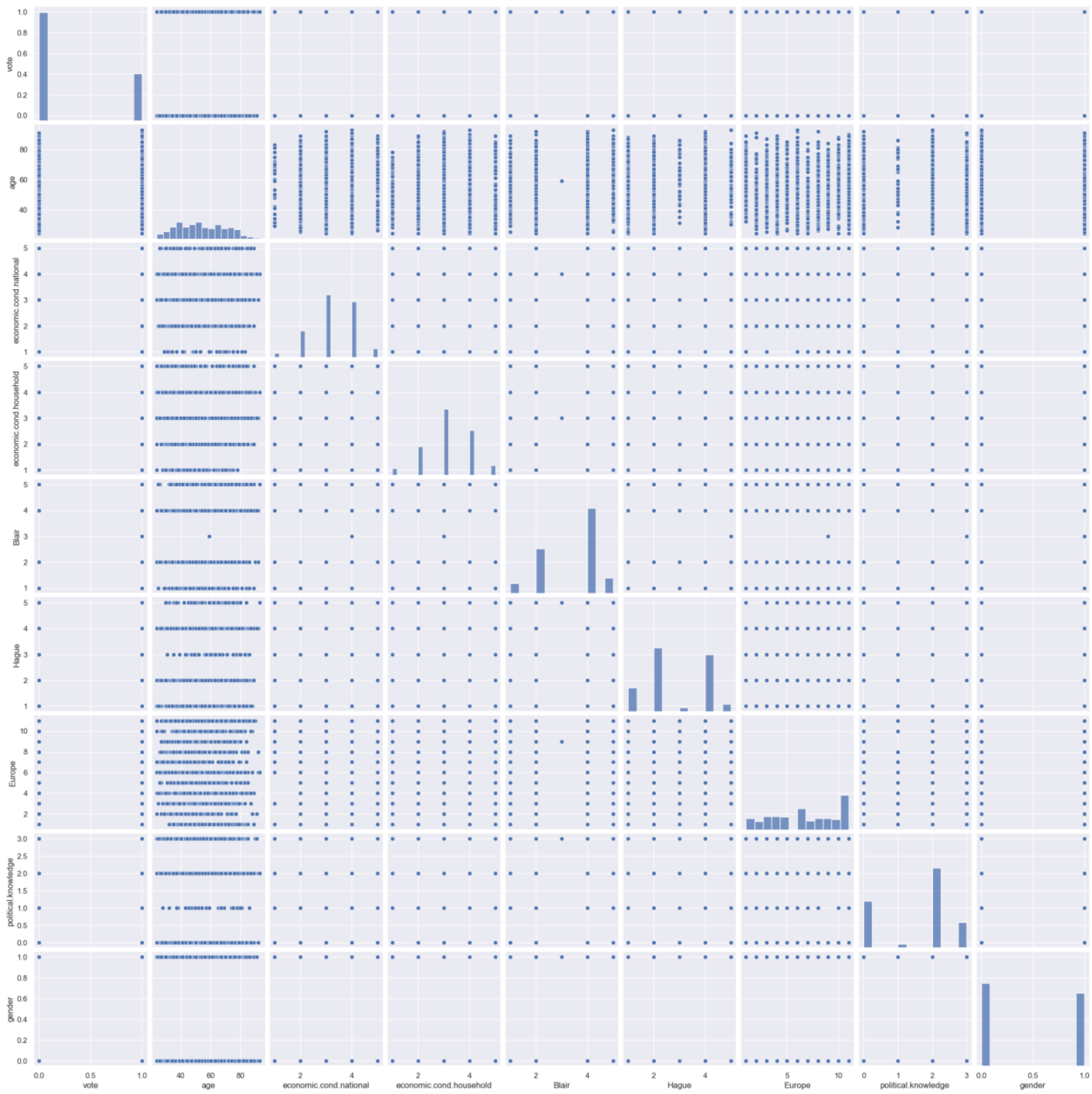
1. **Distribution**:
   - Numerical variables are roughly normally distributed, with some being multimodal.
2. **Outliers**:
   - Outliers are visible in "economic_cond_national" and "economic_cond_household" variables from the boxplots.
3. **Boxplot Limitations**:

- Min and max values are not clear from the boxplots.

Further Analysis Needed:

1. **Identify Outliers**:
   - Use statistical methods (e.g., IQR) to identify and quantify outliers in key variables.
2. **Determine Min and Max Values**:
   - Calculate and report the exact min and max values for all numerical variables.

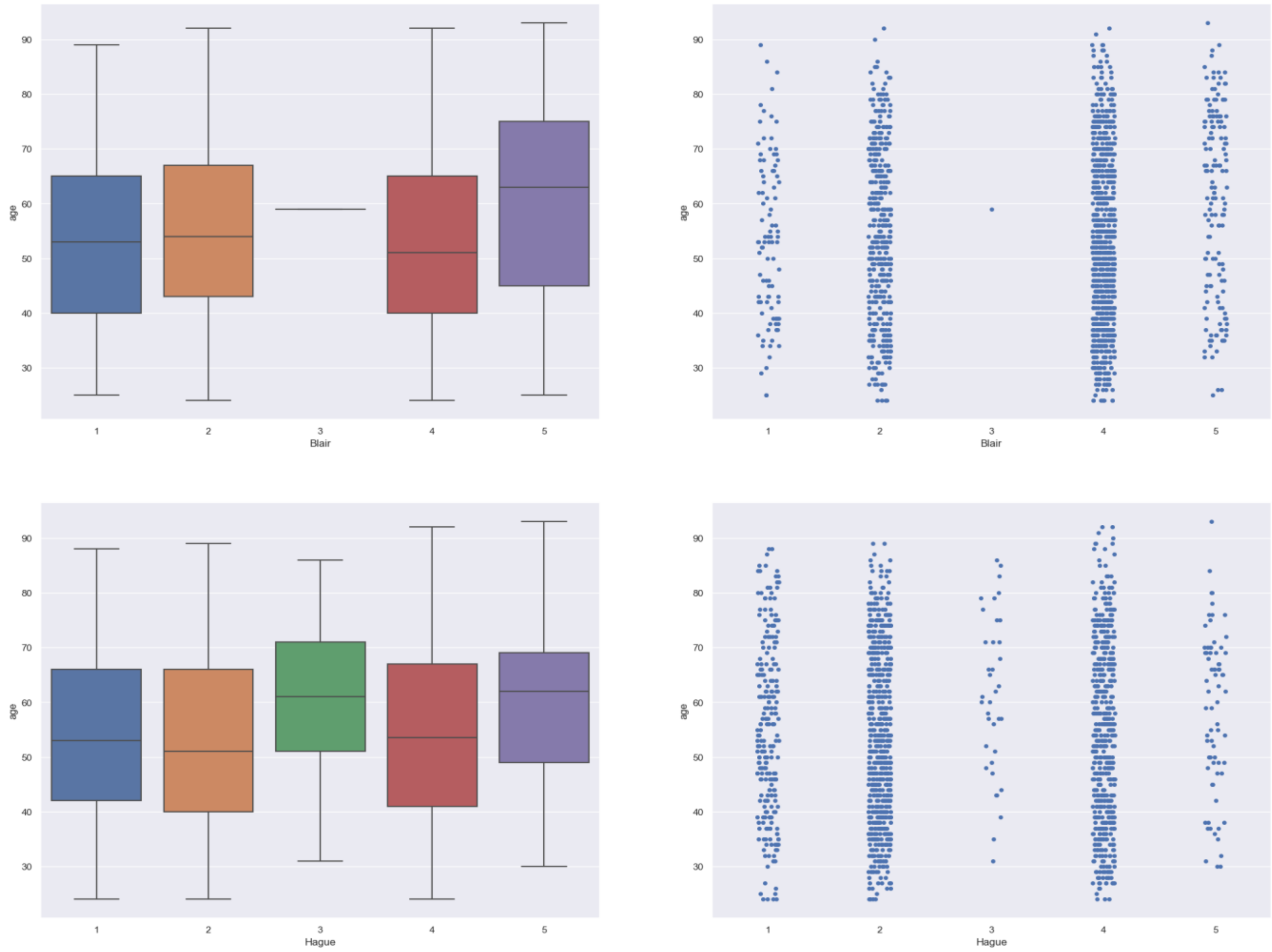
Bivariate Analysis



- **Variable Interactions**:
  - Pairplots show the interaction of each variable with every other variable.
- **Relationship Strength**:
  - No strong relationships are present between the variables.
  - A mixture of positive and negative relationships is observed, as expected.

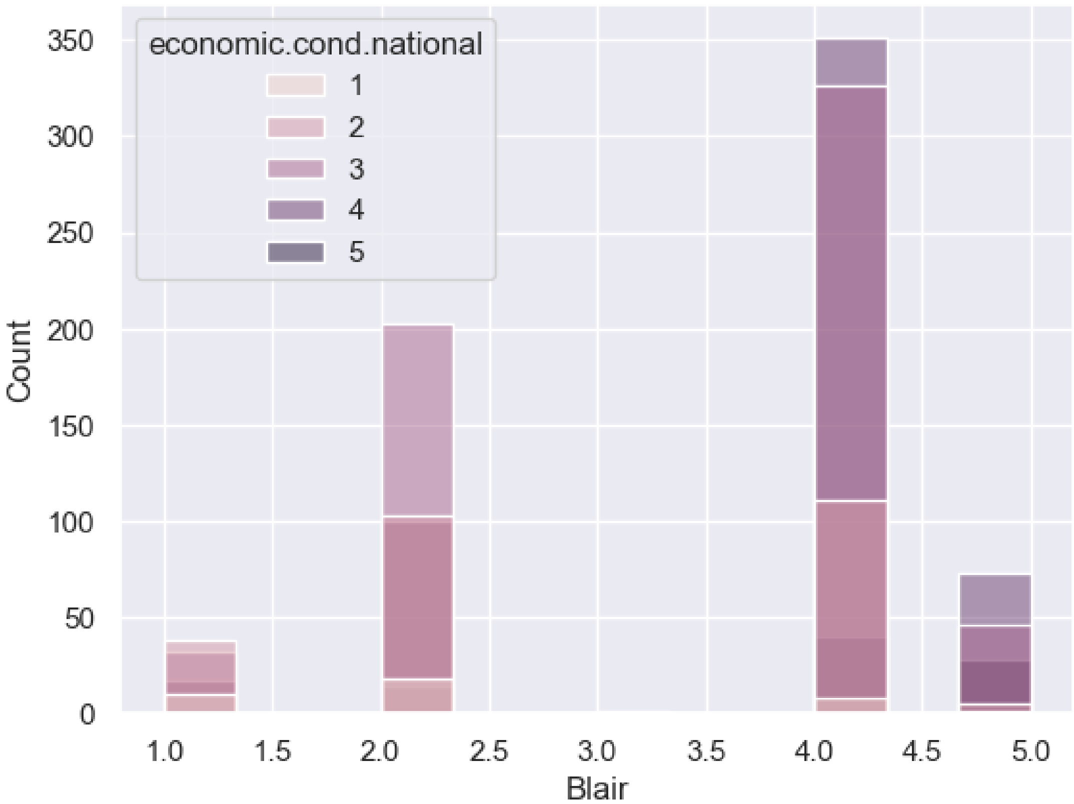
- **Further Analysis**:
  - Overall, pairplots provide a rough estimate of interactions. A clearer picture can be obtained by using heatmaps and different types of plots.

Analysis Blair and Hague



- Hague has slightly more concentration of neutral points than that of Blair for people above 50 years of age.
- People above the age of 45 yrs. generally thinks that Blair is doing a good job

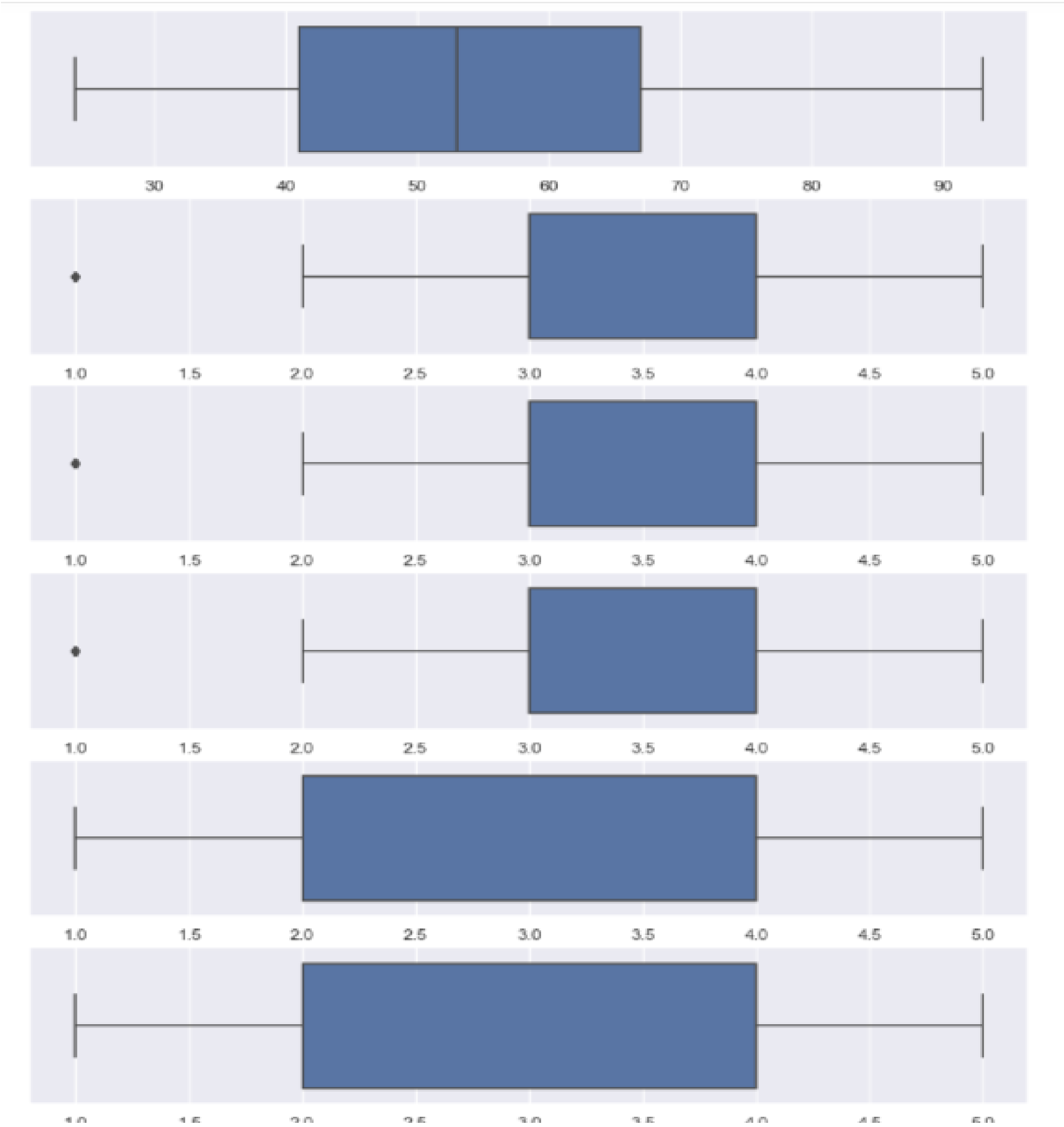
Histplot Analysis - Blair (count) on economic_cond_household.



Histplot Analysis - Hague (count) on economic_cond_household.

Blair has more points in terms of economic national than Hague.

Outliers



There are outliers present in “economic_cond_national” and “economic_cond_household” variables that can be seen from the boxplots. We will find the upper and lower limits to get a clear picture of the outliers.

Heatmap

Observations:

- The correlation matrix shows that most variables have no significant correlation.
- Some variables are moderately positively correlated, while others are slightly negatively correlated.
- 'economic.cond.national' and 'economic.cond.household' have a moderate positive correlation.
- 'Blair' is moderately positively correlated with 'economic.cond.national' and 'economic.cond.household'.

- 'Europe' and 'Hague' have a moderate positive correlation.
- 'Hague' has a moderate negative correlation with 'economic.cond.national' and 'Blair'.
- 'Europe' has a moderate negative correlation with 'economic.cond.national' and 'Blair'.

Train-Test Split:

Our model will use all the variables, with 'vote_Labour' as the target variable. The train-test split technique is used to evaluate the performance of a machine learning algorithm by dividing the dataset into two subsets:

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

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

"We can encode the categorical variables 'vote' and 'gender' into integer types using techniques like Label Encoding or One-Hot Encoding, depending on the requirements of our machine learning model."

Gender Distribution

```
0    812
1    713
Name: gender, dtype: int64
```

Vote Distribution:

```
0    1063
1     462
Name: vote, dtype: int64
```

Since both 'vote' and 'gender' variables have only two classifications, we can use a simple categorical conversion method like pd.Categorical() or dummy encoding with drop_first=True.

This will convert the values into 0 and 1, and since there's no level or order in the subcategories, any encoding method will yield the same result. The resulting datatype after conversion is int8, although converting to int64 is possible but unnecessary as int8 is sufficient.

After Encoding

```
replace = {
    "gender" : {"male" : 1 , "female" : 0},
    "vote"   :  { "Conservative" : 1, "Labour" : 0}
}
```

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
68	0	35	4	4	5	2	3	2	1
627	0	39	3	4	4	2	5	2	1
871	0	38	2	4	2	2	4	3	1
984	1	74	4	3	2	4	8	2	0
1155	1	53	3	4	2	2	6	0	0
1237	0	36	3	3	2	2	6	2	0
1245	0	29	4	4	4	2	2	2	0
1439	0	40	4	3	4	2	2	2	1

Scaling the Data

Scaling the dataset is crucial for bringing data with varying ranges into a similar relative range, optimizing the model's performance. For gradient descent-based algorithms like Linear and Logistic Regression, sensitive to data range, scaling is essential to avoid bias and reduce multicollinearity.

While tree-based methods like Decision Trees don't require scaling due to their split method. We'll scale the 'age' variable using Z-score scaling (standard scaling with mean=0 and standard deviation=1), as other variables are already within a manageable range of 0-10."

Data Splitting:

The data is divided into two subsets: a training set and a testing set. The target variable 'vote_Labour' has been extracted into a separate vector for these subsets. A random state of 1 is chosen to ensure reproducibility.

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

Train-Test-Split Shape:

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

Viewing the data after scaling

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

Apply Logistic Regression and LDA (linear discriminant analysis).

Logistic Regression Model: There are no outliers present in the continuous variable 'age'. The remaining variables are

categorical in nature. Our model will use all the variables and 'vote Labour' is the target variable.

Accuracy - Train data:

```
0.8341187558906692
```

Accuracy - Test data:

```
0.8267543859649122
```

Classification report - Train data:

	precision	recall	f1-score	support
0	0.87	0.91	0.89	735
1	0.77	0.69	0.73	332
accuracy			0.84	1067
macro avg	0.82	0.80	0.81	1067
weighted avg	0.84	0.84	0.84	1067

Linear Discriminant Analysis Model - Observation

Train data:

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

Test data:

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

Model Validity:

- The model does not exhibit signs of overfitting or underfitting.

- The slightly higher error in the test data compared to the train data is acceptable, given the low error margins in both datasets.
- Overall, the model demonstrates good generalization and is appropriately fit to the data.

Apply KNN Model and Naïve Bayes Model.

K-Nearest Neighbor Model: There are no outliers present in the continuous variable 'age'. The remaining variables are categorical in nature. Our modelwill use all the variables and 'vote_Labour' is the target variable. We take K value as 7.

Classification report - Train data:

```
[[649  92]
 [ 86 240]]
              precision    recall  f1-score   support

           0       0.88      0.88      0.88       741
           1       0.72      0.74      0.73       326

    accuracy                           0.83      1067
   macro avg       0.80      0.81      0.80      1067
weighted avg       0.83      0.83      0.83      1067
```

Classification report - Test data:

```
[[284  36]
 [ 44  94]]
              precision    recall  f1-score   support

           0       0.87      0.89      0.88       320
           1       0.72      0.68      0.70       138

    accuracy                           0.83       458
   macro avg       0.79      0.78      0.79       458
weighted avg       0.82      0.83      0.82       458
```

K-Nearest Neighbor Model - Observation

Train data:

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

Test data:

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

Model Validity:

- The model is overfitting the training data.
- There is a significant drop in accuracy (more than 10%) from the training data (100%) to the test data (83.77%).
- This difference indicates that the K-Nearest Neighbor (KNN) model is not generalizing well and is fitting too closely to the training data.

Naïve Bayes Model – Observation

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

Classification report - Train data:

```
[[649  92]
 [ 86 240]]
              precision    recall  f1-score   support

           0       0.88      0.88      0.88       741
           1       0.72      0.74      0.73       326

    accuracy                           0.83      1067
   macro avg       0.80      0.81      0.80      1067
weighted avg       0.83      0.83      0.83      1067
```

Classification report - Test data:

```
[[284  36]
 [ 44  94]]
              precision    recall  f1-score   support

           0       0.87      0.89      0.88       320
           1       0.72      0.68      0.70       138

    accuracy                           0.83       458
   macro avg       0.79      0.78      0.79       458
weighted avg       0.82      0.83      0.82       458
```

Naïve Bayes Model - Observation

Train data:

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

Test data:

- Accuracy: 82.24%
- Precision: 87%
- Recall: 87%
- F1-Score: 88%

Model Validity:

- The model is not exhibiting signs of overfitting or underfitting.
- Although there is a slight increase in error in the test data compared to the train data, the error margin is low and acceptable.
- Both the train and test data errors are reasonably close, indicating that the model generalizes well and is appropriately fit to the data.

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

Tuning is crucial for maximizing a model's performance without introducing overfitting or excessive variance. In machine learning, this is achieved by adjusting hyperparameters appropriately.

Grid Search is a widely used method for parameter optimization. It involves defining a set of parameters and evaluating the performance of each parameter combination using cross-validation. The best-performing parameter combination is then selected.

Models such as Bagging, Boosting, Gradient boosting, Cat boosting, etc., are susceptible to underfitting or overfitting the data. Overfitting occurs when a model performs exceptionally well on the training data but poorly on the test data. Underfitting, on the other hand, means the model performs well on the test data but poorly on the training data. Finding the right balance to avoid these issues is essential for building a robust and accurate machine learning model.

Bagging Model using Random Forest Classifier:

Bagging is an ensemble technique that combines multiple base models to create an optimal model. It's particularly effective with tree-based algorithms and is designed to enhance the performance of machine learning algorithms in classification or regression tasks. Each base classifier is trained independently with a training set generated by randomly selecting data from the training set with replacement.

```
GridSearchCV(cv=10, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [5, 10, 15, 20],
                         'min_samples_leaf': [15, 25, 35, 50],
                         'min_samples_split': [30, 50, 70, 100],
                         'random_state': [0]})
```

In this case, we'll use Random Forest as the base classifier for bagging. The hyperparameters used in the model are:

- max_depth
- max_features
- min_samples_leaf
- min_samples_split
- n_estimators

After performing GridSearchCV, the best parameters obtained are:

- 'max_depth': 5
- 'max_features': 7
- 'min_samples_leaf': 25
- 'min_samples_split': 60
- 'n_estimators': 101

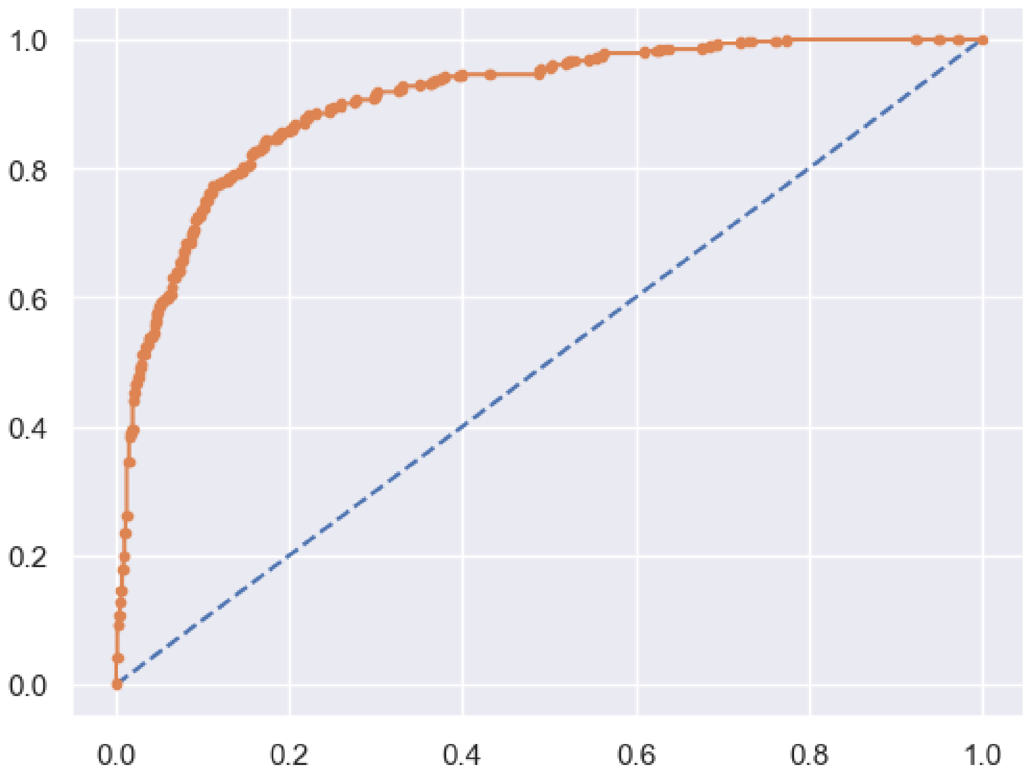
These optimized parameters are chosen to maximize the model's performance while avoiding overfitting and high variance.

```
0.8388003748828491
[[687  48]
 [124 208]]
              precision    recall  f1-score   support

           0       0.85      0.93      0.89       735
           1       0.81      0.63      0.71       332

    accuracy                           0.84      1067
   macro avg       0.83      0.78      0.80      1067
weighted avg       0.84      0.84      0.83      1067
```

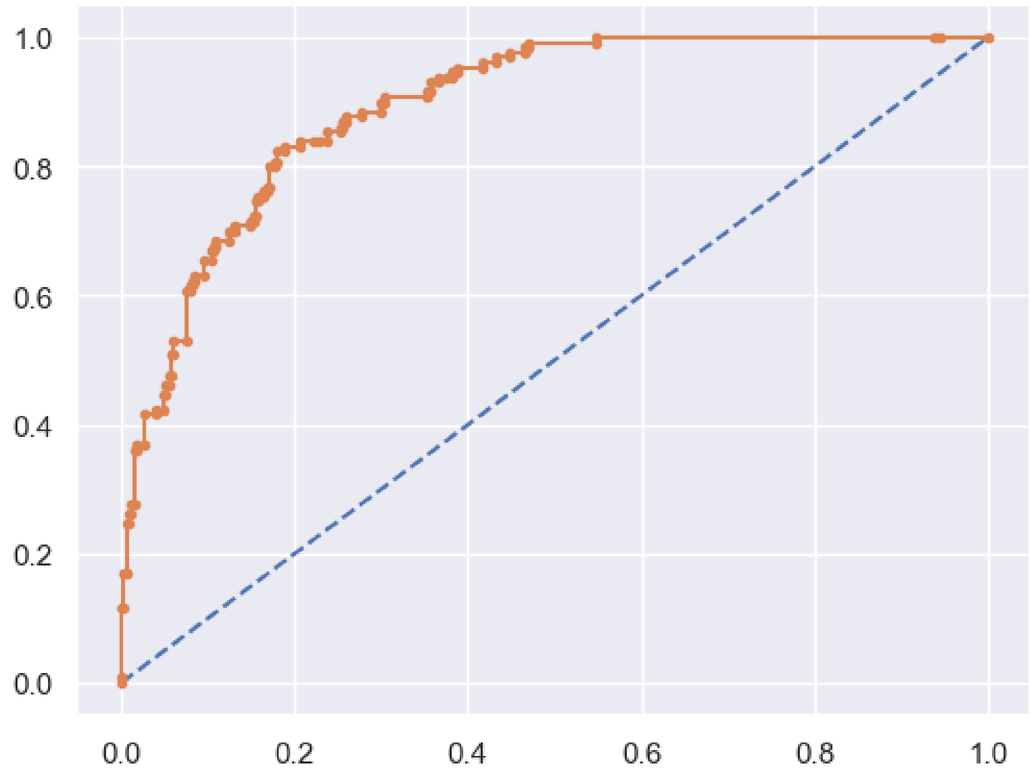
Now the results for unscaled data

```
Train Accuracy- 0.8303655107778819
Test Accuracy- 0.834061135371179
```

AUC and ROC for the training data



calculate AUC



Boosting Model:

Boosting is another ensemble technique that transforms weak learners into strong learners by sequentially combining their results. Unlike bagging, boosting is a sequential method where each weak learner's prediction becomes the input for the next learner, improving the overall model performance.

In boosting, each application of the base learning algorithm generates a new weak learner prediction rule. This iterative process combines these weak rules into a single strong prediction rule. Misclassified data points gain higher weights, while correctly classified examples lose weight. This way, future weak learners focus more on the examples that previous learners struggled with.

Boosting techniques are also tree-based methods. For this project, the following boosting techniques will be used:

1. ADA Boost (Adaptive Boosting)
2. Gradient Boosting
3. Extreme Gradient Boosting (XGBoost)
4. CAT Boost (Categorical Boosting)

These boosting techniques help in improving the model's performance by effectively combining weak learners into a strong prediction rule. Each technique has its own unique approach to boosting and can be applied based on the specific requirements of the project.

ADA Boost Model

For the AdaBoost model, it's known for enhancing binary classifiers and now extends to improving multiclass classifiers too. It acts as a corrective layer on top of any classifier, earning its reputation as the "best out-of-the-box classifier." Key hyperparameters for AdaBoost include 'algorithm' and 'n_estimators'. While other parameters exist, we'll focus on these for grid search, using default values for the rest.

After conducting GridSearchCV, the optimal parameters are:

- 'algorithm' = 'SAMME'
- 'n_estimators' = 50

Results for unscaled data are as follows:

- Train Accuracy: 0.8369
- Test Accuracy: 0.8428

```
0.8350515463917526
[[674  61]
 [115 217]]
              precision    recall  f1-score   support

           0       0.85      0.92      0.88       735
           1       0.78      0.65      0.71       332

    accuracy                           0.84      1067
   macro avg       0.82      0.79      0.80      1067
weighted avg       0.83      0.84      0.83      1067
```

Gradient Boosting model

The Gradient Boosting model, similar to AdaBoost, improves by sequentially addressing misidentified predictors and under-fitted predictions. The key distinction is in handling misidentified values from previous weak learners. Gradient Boosting fits new predictors to residual errors from previous ones, progressively refining the model.

For model building, essential hyperparameters include:

- Criterion
- Loss
- n_estimators
- max_features
- min_samples_split

After GridSearchCV, the optimal parameters are:

- 'criterion' = 'friedman_mse'
- 'loss' = 'exponential'
- 'n_estimators' = 50
- 'max_features' = 8
- 'min_samples_split' = 45

	0	1
0	0.179738	0.820262
1	0.016951	0.983049
2	0.026527	0.973473
3	0.187475	0.812525
4	0.001633	0.998367
5	0.051704	0.948296
6	0.039480	0.960520
7	0.035373	0.964627
8	0.018253	0.981747
9	0.679936	0.320064

XGBoost (eXtreme Gradient Boosting) Model:

XGBoost is a model based on the gradient boosting framework but with significant improvements in performance through systems optimization and algorithmic enhancements. It utilizes parallel processing and RAM optimizations to enhance the Gradient Boost method to its peak, earning the name "extreme."

One advantage of XGBoost is its automatic handling of null values by using the parameter "missing=NaN." Additionally, XGBoost does not contain the parameter 'min_sample_split,' distinguishing it from traditional Gradient Boosting.

Before fitting the XGBoost model, it's important to understand the hyperparameters involved in model building, which include:

- max_depth
- min_samples_leaf
- n_estimators
- learning_rate

After performing GridSearchCV, the best parameters obtained for the XGBoost model are:

- 'max_depth': 4
- 'min_samples_leaf': 15
- 'n_estimators': 50
- 'learning_rate': 0.1

These optimized parameters are selected to maximize the model's performance and accuracy. XGBoost's enhanced features and optimized parameters make it a powerful tool for predictive modeling tasks.

Bagging

Before Scaling

Train Accuracy- 0.8303655107778819

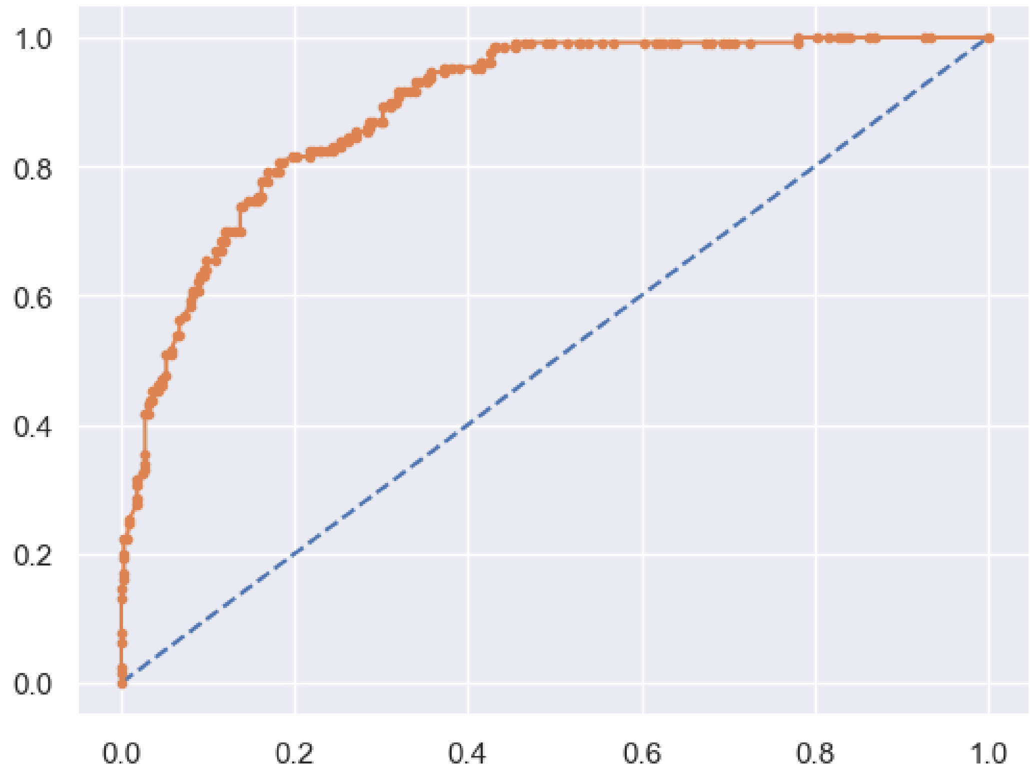
Test Accuracy- 0.834061135371179

```
0.8350515463917526
[[674  61]
 [115 217]]
              precision    recall  f1-score   support

           0       0.85      0.92      0.88       735
           1       0.78      0.65      0.71       332

    accuracy                           0.84      1067
   macro avg       0.82      0.79      0.80      1067
weighted avg       0.83      0.84      0.83      1067
```

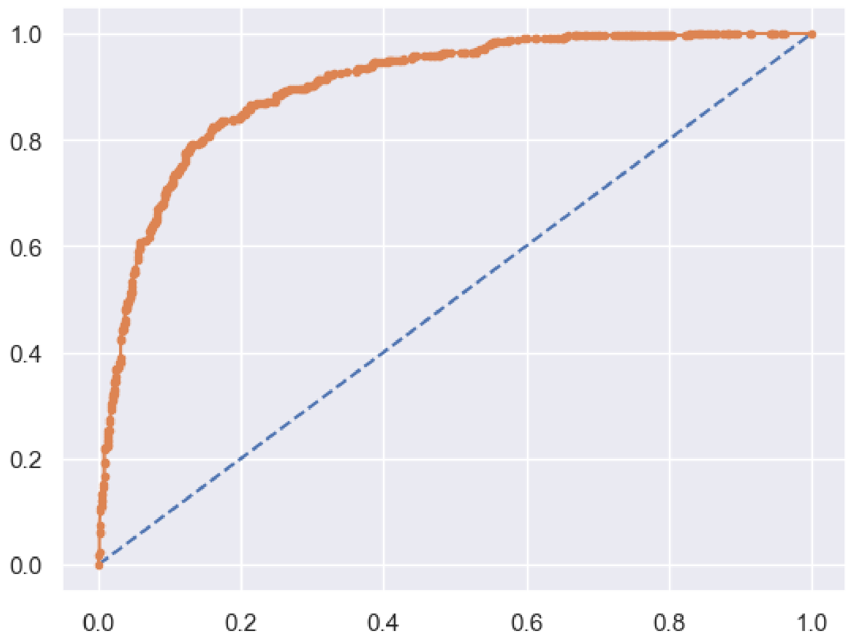
```
0.8318777292576419
[[296  32]
 [ 45  85]]
              precision    recall  f1-score   support

           0       0.87      0.90      0.88       328
           1       0.73      0.65      0.69       130

    accuracy                           0.83       458
   macro avg       0.80      0.78      0.79       458
weighted avg       0.83      0.83      0.83       458
```

Area Under ROC Curve and AUC Score: For both Training and Testing:

AUC = 0.893



AUC =0.902



After Scaling

Train Accuracy- 0.8303655107778819

Test Accuracy- 0.834061135371179

Confusion Matrix

For Train Data For Test Data True Negative: 201 False Positive: 122 True Negative: 83 False Positive: 56 False Negative: 59 True Positive: 685 False Negative: 20 True Positive: 299

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.

Performance Metrics: Various performance metrics are utilized to evaluate the robustness of a model and comprehend its performance, aiding in informed decisions regarding its real-time application. Industry standards typically rely on the following methods:

- Classification Accuracy
- Confusion Matrix
- Classification Report
- Area Under ROC Curve (visualization) and AUC Score

These metrics provide insights into different aspects of a model's performance, ranging from overall accuracy to its ability to correctly classify different classes, and its capacity to distinguish between classes through the ROC curve and AUC score.

Logistic Regression

Before Scaling Train Accuracy- 0.8303655107778819

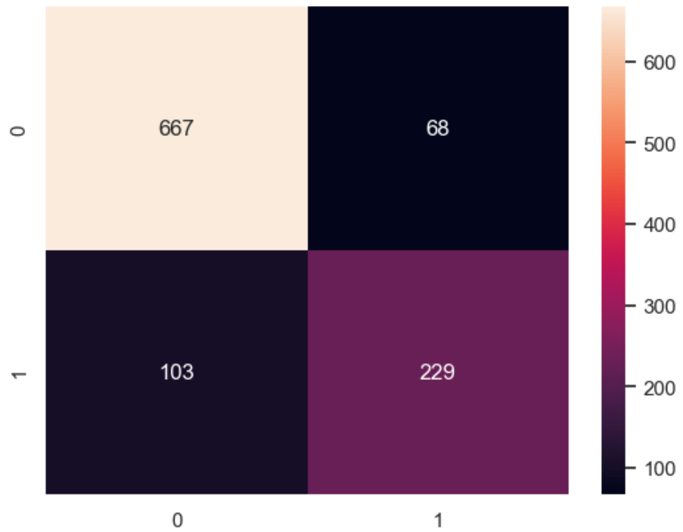
Test Accuracy- 0.8537117903930131

Confusion Matrix

For Train Data For Test Data

True Negative: 212 False Positive: 111 True Negative: 94 False Positive: 45

False Negative: 70 True Positive: 674 False Negative: 22 True Positive: 297



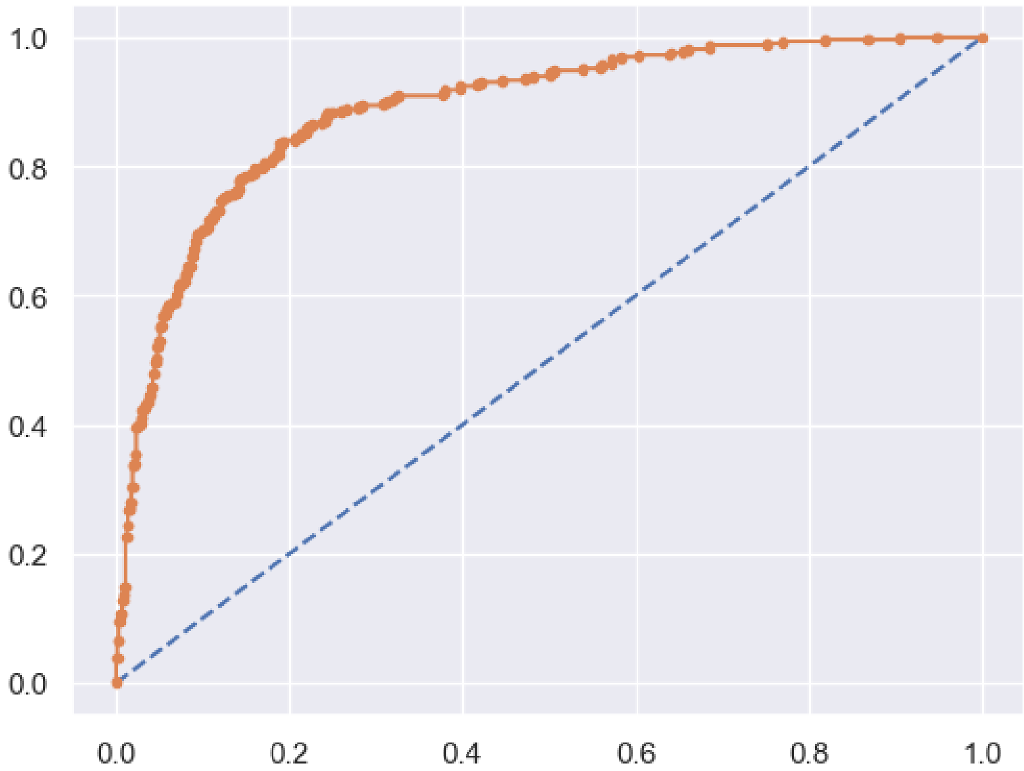
Classification Report

	precision	recall	f1-score	support
0	0.87	0.91	0.89	735
1	0.77	0.69	0.73	332
accuracy			0.84	1067
macro avg	0.82	0.80	0.81	1067
weighted avg	0.84	0.84	0.84	1067

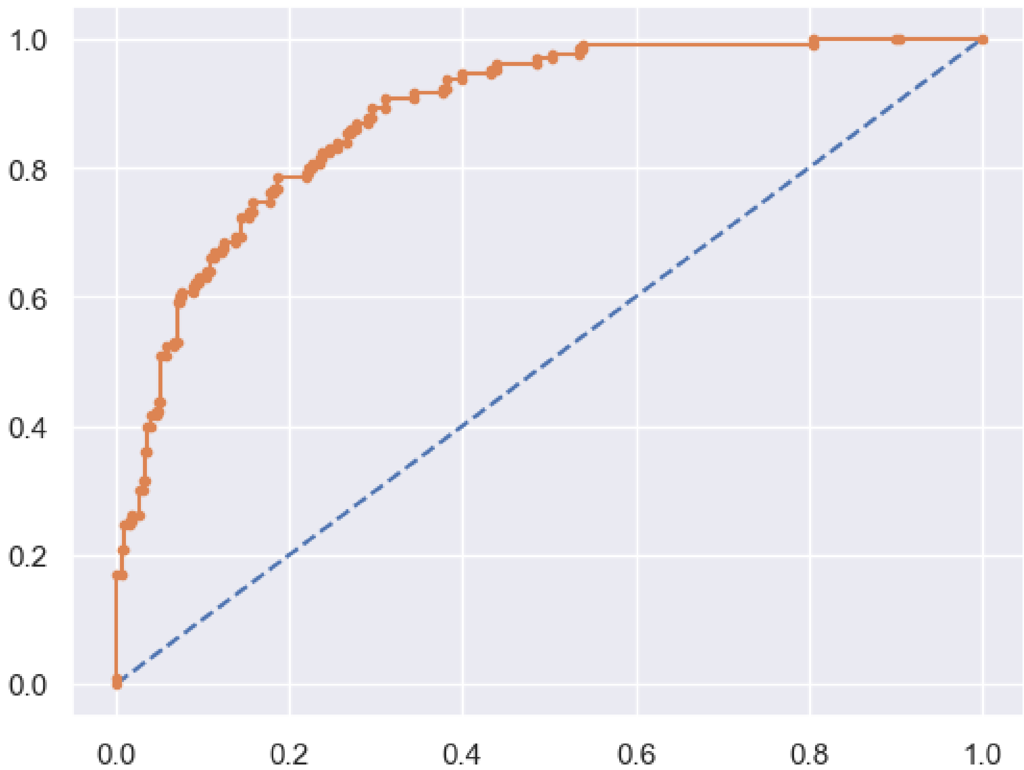
Area Under ROC Curve and AUC Score:

For both Training and Testing:

AUC and ROC for the training data



AUC and ROC for the test data



Confusion Matrix

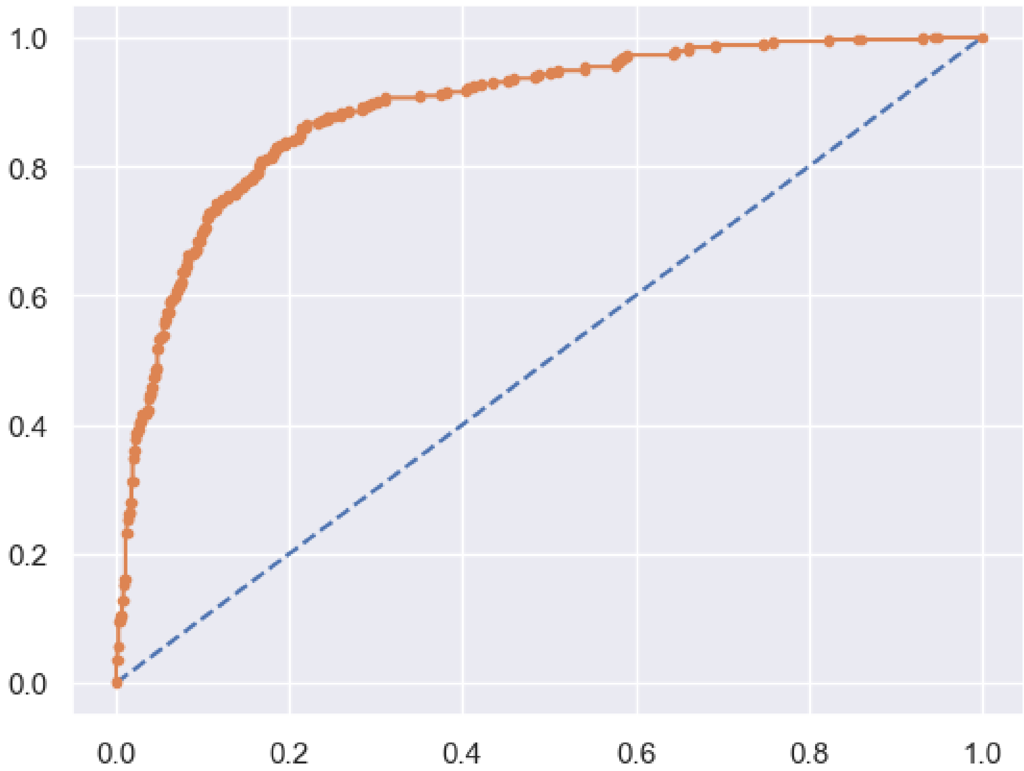
Classification Report

```
[[289  44]
 [ 39  86]]
              precision    recall  f1-score   support

           0       0.88      0.87      0.87       333
           1       0.66      0.69      0.67       125

    accuracy                           0.82       458
   macro avg       0.77      0.78      0.77       458
weighted avg       0.82      0.82      0.82       458
```

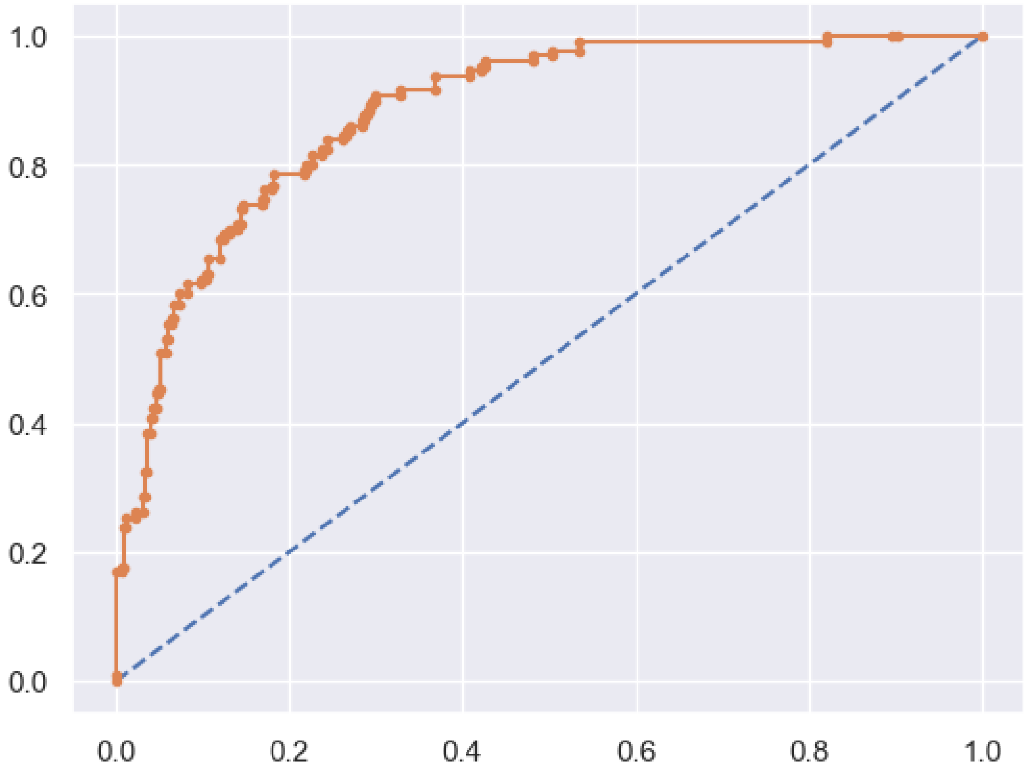
Train Test

```
[[660  99]
 [ 75 233]]
              precision    recall  f1-score   support

           0       0.90      0.87      0.88       759
           1       0.70      0.76      0.73       308

    accuracy                           0.84      1067
   macro avg       0.80      0.81      0.81      1067
weighted avg       0.84      0.84      0.84      1067
```

AUC and ROC for the training data



AUC and ROC for the test data



Insights:

- The Labour party has more than double the votes of the Conservative party.
- Most people rated the national economic condition between 3 and 4, with an average score of 3.245221. Similarly, for the household economic condition, most ratings fell between 3 and 4, with an average score of 3.137772.
- Blair received more votes than Hague, with significantly better scores for Blair (average score 3.335531) compared to Hague (average score 2.749506).
- Approximately 30% of the population has minimal political knowledge (score of 0 on a scale of 0 to 3).
- Despite giving low scores to a party, some voters still chose to vote for that party, possibly due to limited political knowledge.
- Eurosceptic sentiment influenced voting, with higher Euroscepticism correlating with votes for the Conservative party and lower Euroscepticism correlating with votes for the Labour party.
- Among those with no political knowledge (score of 0), the majority voted for the Labour party.
- Tuned models performed better than regular models, with Gradient Boosting (tuned) identified as the best/optimized model.

Business Recommendations:

- Hyper-parameter tuning is crucial for model building, although it requires significant processing power.
- Gathering more data can enhance model training and predictive accuracy.
- Sequentially predicting outcomes using all models can provide a better understanding of probability and outcomes.
- Using the Gradient Boosting model without scaling is recommended due to its optimized performance.

Problem 2

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

President Franklin D. Roosevelt in 1941

President John F. Kennedy in 1961

President Richard Nixon in 1973

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

```
Speech by President Roosevelt (1941):
Number of characters: 7571
Number of words: 1526
Number of sentences: 68

Speech by President Kennedy (1961):
Number of characters: 7618
Number of words: 1543
Number of sentences: 52

Speech by President Nixon (1973):
Number of characters: 9991
Number of words: 2006
Number of sentences: 68
```

Characters

- President Franklin D. Roosevelt's speech: 7571 characters
- President John F. Kennedy's speech: 7618 characters

- President Richard Nixon's speech: 9991 characters

Words

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

Sentences

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

Remove all the stop-words from all three speeches

Three most common words used in all three speeches:

```
Word count after stop-word removal:
• President Franklin D. Roosevelt's speech has 617 words.
• President John F. Kennedy's speech has 684 words.
• President Richard Nixon's speech has 822 words.

Most common words after stop-word removal:
• President Franklin D. Roosevelt's most common words: [('nation', 12), ('know', 10), ('spirit', 9)]
• President John F. Kennedy's most common words: [('let', 16), ('us', 12), ('world', 8)]
• President Richard Nixon's most common words: [('us', 26), ('let', 22), ('america', 21)]
```

Problem 2 - Plot Word cloud of all three speeches

Word Cloud for President Roosevelt's Speech

Word Cloud for President Kennedy's Speech

Word Cloud for President Nixon's Speech

Most common in all three speeches

Most Common Words in All Three Speeches