



Machine Learning Project

Issac Abraham

Table of Contents

Decoding Voter Preferences: A Data-Driven Approach to Election Strategy Optimization.....	4
Executive Summary.....	4
Problem 1.....	4
DATA DICTIONARY.....	4
Dataset Ingestion and Overview.....	4
Sample of the dataset.....	5
Data Overview and Types.....	5
Descriptive Statistics.....	5
Null Value Check.....	6
Checking for Duplicate Rows.....	6
Skewness Analysis of Numerical Variables.....	6
Data Types Overview.....	7
Dataset Shape Overview.....	8
Univariate Analysis: Distribution of Continuous Variables.....	8
Outlier Detection: Box Plots for Continuous Variables.....	9
Count Plots for Categorical Variables: Gender and Vote.....	9
Bivariate Analysis: Pair Plot for Vote and Variable Relationships.....	9
Encoding Categorical Variables.....	11
Is Scaling Necessary for Features?	11
Data Split and Shape Display.....	11
Logistic Regression Model Evaluation.....	11
K-Nearest Neighbors (KNN) Model Evaluation.....	12
Naïve Bayes Model Evaluation.....	13
GridSearchCV.....	14
Logistic Regression with GridSearchCV.....	14
Linear Discriminant Analysis (LDA) with GridSearchCV.....	15
K-Nearest Neighbors (KNN) with GridSearchCV.....	15
Naive Bayes with GridSearchCV.....	16
Bagging with Random Forest.....	17
GridSearchCV for Random Forest.....	17
Random Forest Performance.....	17
Boosting with AdaBoost and Random Forest.....	17
AdaBoost with Random Forest.....	17
ROC Curve and AUC Scores for models.....	18
Conclusion: Business insights and Recommendations.....	22
Analyzing Presidential Inaugural Speeches: Insights and Trends.....	22
Problem 2.....	23
Introduction.....	23
NLTK (Natural Language Toolkit)	23
Matplotlib (for future visualizations)	23
Characters, Words and Sentences in the Speeches.....	23

Interpretation of Results.....	23
Removing stop words:	
Most Frequent Words in Presidential Inaugural Addresses.....	24
Word Clouds from Inaugural Speeches.....	25

List of Figures

Fig 1: Sample of dataset.....	5
Fig 2: Datatypes.....	5
Fig 3: Descriptive Statistics.....	6
Fig 4: Null Value.....	6
Fig 5: Skewness Values.....	7
Fig 6: Data types overview.....	8
Fig 7 : Distribution plots.....	8
Fig 8: Box Plots.....	8
Fig 9 : Count plots.....	9
Fig 10 : Pairplots.....	10
Fig 11: Data shape output.....	11
Fig 12: Performance metrics(Logistic Regression)	12
Fig 13: Performance metrics(K -Nearest Neighbours)	13
Fig 14: Performance metrics: Naïve bayes model.....	14
Fig 15: Logistic regression GridSearchCV.....	15
Fig 16: LDA GridsearchCV.....	15
Fig 17: KNN GridSearchCV	16
Fig 18: Naïve Bayes GridSearchCV.....	16
Fig 19: ROC Logistic regression.....	18
Fig 20: ROC LDA.....	18
Fig 21: ROC KNN.....	19
Fig 22: ROC Naïve Bayes.....	19
Fig 23:ROC Bagging.....	20
Fig 24: ROC Adaboosting.....	20
Fig 25: Output-Character words and speeches.....	24
Fig 26: Word counts after removing stop words.....	24
Fig 27:Most Frequent words.....	25
Fig 28:Word Cloud.....	26

Decoding Voter Preferences: A Data-Driven Approach to Election Strategy Optimization

Executive Summary

CNBE's latest election data analysis is thoroughly examined in this report, which also uses machine learning algorithms to forecast voter behaviour. The aim is to provide valuable perspectives on the major determinants of voter behaviour and suggest industry-specific election campaign tactics.

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.

DATA DICTIONARY:

1. **vote:** Political party chosen by the voter.
2. **age:** Age of the voter.
3. **economic.cond.national:** Voter's perception of the national economic condition.
4. **economic.cond.household:** Voter's perception of the household economic condition.
5. **Blair:** Voter's preference rating for the Blair party.
6. **Hague:** Voter's preference rating for the Hague party.
7. **Europe:** Voter's stance on European issues.
8. **political.knowledge:** Level of political knowledge of the voter.
9. **gender:** Gender of the voter.

We begin our study by initialising the necessary Python packages to enable a thorough analysis of election data. Seaborn, Matplotlib, and Pandas are used to make data processing and visualisation easy. Moreover, scikit-learn offers a strong framework for applying machine learning algorithms, guaranteeing a data-driven method for interpreting voter preferences.

Dataset Ingestion and Overview

Data from the transformed 'Election_Data.csv' file (which was previously an XLSX file including multiple sheets) is imported into a Pandas DataFrame called 'df.' The first 10 rows are shown to give a brief summary of the data structure and a look at the important variables and their values.

Sample 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
5	6	Labour	47	3	4	4	4	4	2	male
6	7	Labour	57	2	2	4	4	11	2	male
7	8	Labour	77	3	4	4	1	1	0	male
8	9	Labour	39	3	3	4	4	11	0	female
9	10	Labour	70	3	2	5	1	11	2	male

Fig 1: Sample of dataset

Data Overview and Types

Using `df.info()`, we can see that there are 1525 entries in the dataset spread over 10 columns. For quantitative variables like age and political knowledge, the data types include integers (int64), and for qualitative variables like vote and gender, the data types are objects.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 10 columns):
#   Column              Non-Null Count  Dtype
---  ---
0   Unnamed: 0          1525 non-null   int64
1   vote                1525 non-null   object
2   age                 1525 non-null   int64
3   economic.cond.national 1525 non-null   int64
4   economic.cond.household 1525 non-null   int64
5   Blair               1525 non-null   int64
6   Hague               1525 non-null   int64
7   Europe              1525 non-null   int64
8   political.knowledge  1525 non-null   int64
9   gender              1525 non-null   object
dtypes: int64(8), object(2)
memory usage: 119.3+ KB
```

Fig 2: Datatypes

Descriptive Statistics

The `df.describe()` output provides insightful descriptive statistics for the numeric variables in the dataset. The mean, standard deviation, and quartile values offer a snapshot of the central tendency, variability, and distribution of variables such as age, economic perceptions, and political knowledge. The average voter age (mean = 54.18), the range of economic perspectives, and the degree of political awareness are important points to note.

	Unnamed: 0	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge
count	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000
mean	763.000000	54.182295	3.245902	3.140328	3.334426	2.746885	6.728525	1.542295
std	440.373894	15.711209	0.880969	0.929951	1.174824	1.230703	3.297538	1.083315
min	1.000000	24.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000
25%	382.000000	41.000000	3.000000	3.000000	2.000000	2.000000	4.000000	0.000000
50%	763.000000	53.000000	3.000000	3.000000	4.000000	2.000000	6.000000	2.000000
75%	1144.000000	67.000000	4.000000	4.000000	4.000000	4.000000	10.000000	2.000000
max	1525.000000	93.000000	5.000000	5.000000	5.000000	5.000000	11.000000	3.000000

Fig 3: Descriptive Statistics

Null Value Check

Executing 'df.isnull().sum()' confirms that there are no missing values in the dataset. Each column, including 'Unnamed: 0', 'vote', 'age', and others, exhibits a count of zero null entries, ensuring the dataset's completeness. This observation ensures that the dataset is intact, which makes it easier to conduct a thorough analysis without having to deal with missing values or imputation.

Unnamed: 0	0
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	

Fig 4: Null Value

Checking for Duplicate Rows

In this step, we examined the dataset for any duplicate rows. Duplicates can occur when the same set of values appears more than once in the dataset. The result, "Number of duplicate rows = 0," indicates that there are no duplicate rows, ensuring data integrity for subsequent analyses.

Distribution Analysis of Categorical Variables

There are two distinct values in the "Vote" column: "Conservative" and "Labour." Voters are divided 462 times in favour of "Conservative" and 1063 times in favour of "Labour." This suggests an unequal distribution, with a large majority of "Labour" votes cast compared to "Conservative" ballots.

With "Male" and "Female" as the two unique values, the "Gender" column also displays a binary distribution in terms of gender. The data set has 713 occurrences of "Male" and 812 instances of "Female," indicating a fairly even representation of gender.

Our research requires an understanding of the distribution of these category variables, particularly in light of the unbalanced structure of the vote distribution. During the modelling process, this knowledge will be helpful for analysing the findings and coming to wise judgements.

Skewness Analysis of Numerical Variables

Analysing the skewness of numerical variables sheds light on the dataset's distributional properties. A distribution's asymmetry is measured by its skewness. The skewness values for important numerical variables are as follows:

Age: The distribution is very symmetrical, as the skewness is almost zero (0.145).

Economic Condition (National): There is a minor leftward skewness shown by the negative (-0.240) skewness, which indicates that more responses are focused on the higher end of the scale.

The Household Economic situation has a similar leftward skewness, but one that is not as prominent as the national economic situation. The skewness is negative (-0.150).

Blair: The skewness indicates a leftward skewness, with more respondents leaning towards lower values. It is negative (-0.535).

Hague: There is a small rightward skewness, with a positive skewness of 0.152, indicating that more respondents are inclined towards higher values.

Europe: The distribution appears to be somewhat symmetrical, since the skewness is about zero (-0.136).

Political Knowledge: A leftward skewness is shown by the negative (-0.427) skewness, which indicates that more respondents possess a greater level of political knowledge.

```
Skewness of Numerical Variables:
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
dtype: float64
```

Fig 5: Skewness Values

Data Types Overview

The data types for every column in the dataset are shown in the output.

```

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

```

Fig 6: Data types overview

Dataset Shape Overview

The dataset contains 1525 rows and 10 columns, representing the dimensions of the dataset.

Univariate Analysis: Distribution of Continuous Variables

The distribution plots display the distribution of key continuous variables, providing insights into their patterns and spread in the dataset.

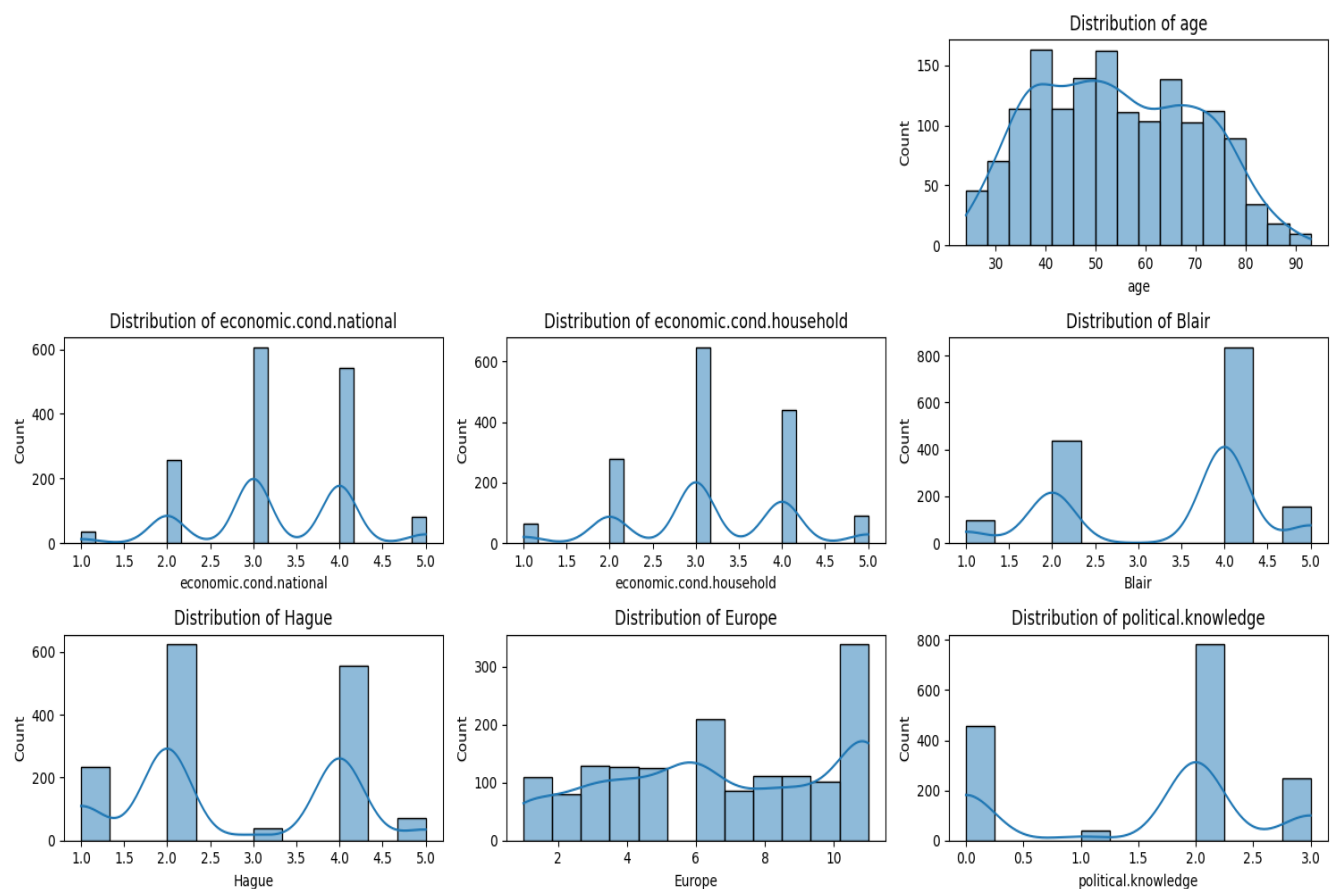


Fig 7 : Distribution plots

Outlier Detection: Box Plots for Continuous Variables

We don't see a significant number of outlier so we are not treating it in this case study.

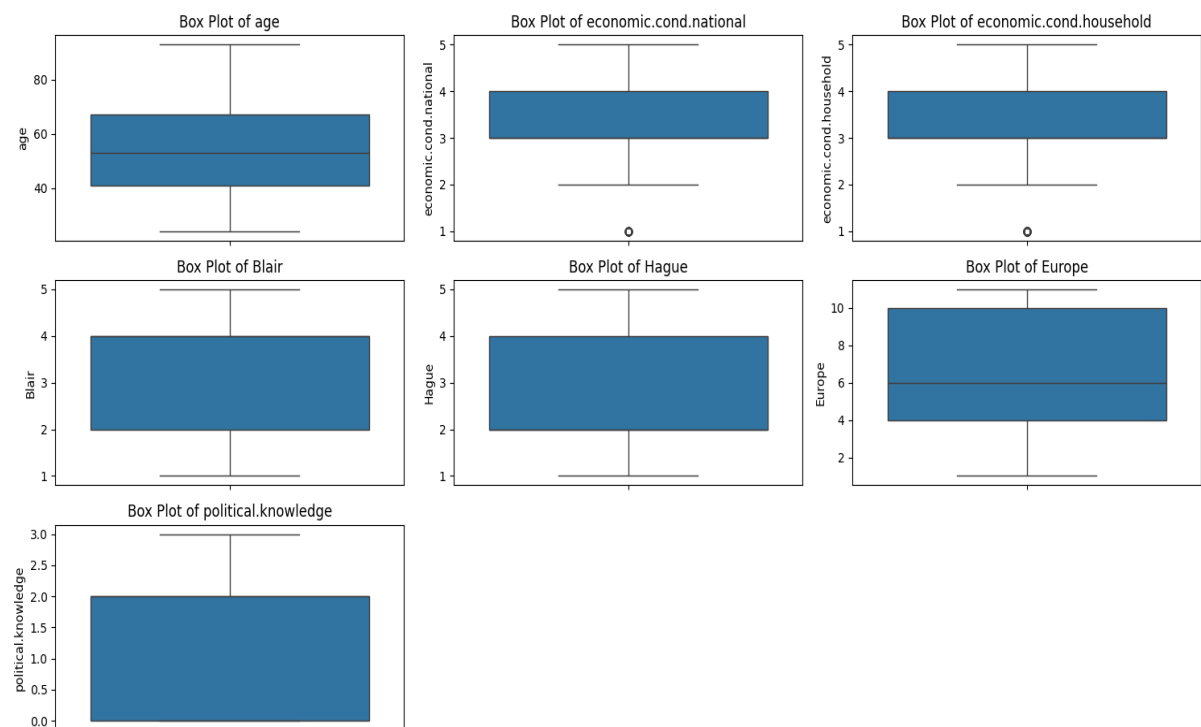


Fig 8: Box Plots

Count Plots for Categorical Variables: Gender and Vote

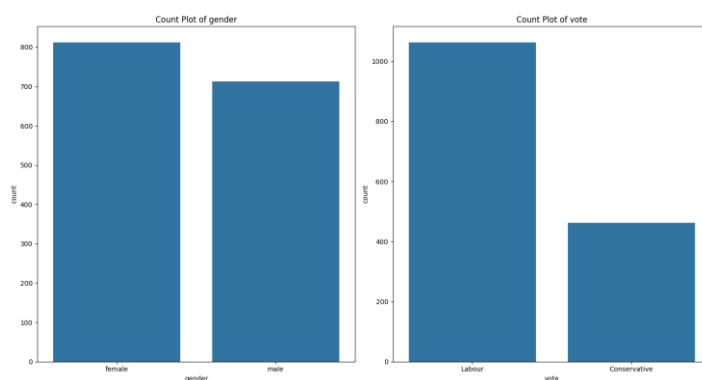


Fig 9 : Count plots

These count plots provide a visual representation of the distribution of gender and vote categories, helping to understand the frequency of each class within the dataset.

Bivariate Analysis: Pair Plot for Vote and Variable Relationships

The pair plot visually represents the relationships between variables concerning the 'vote' category, offering insights into potential patterns and correlations.

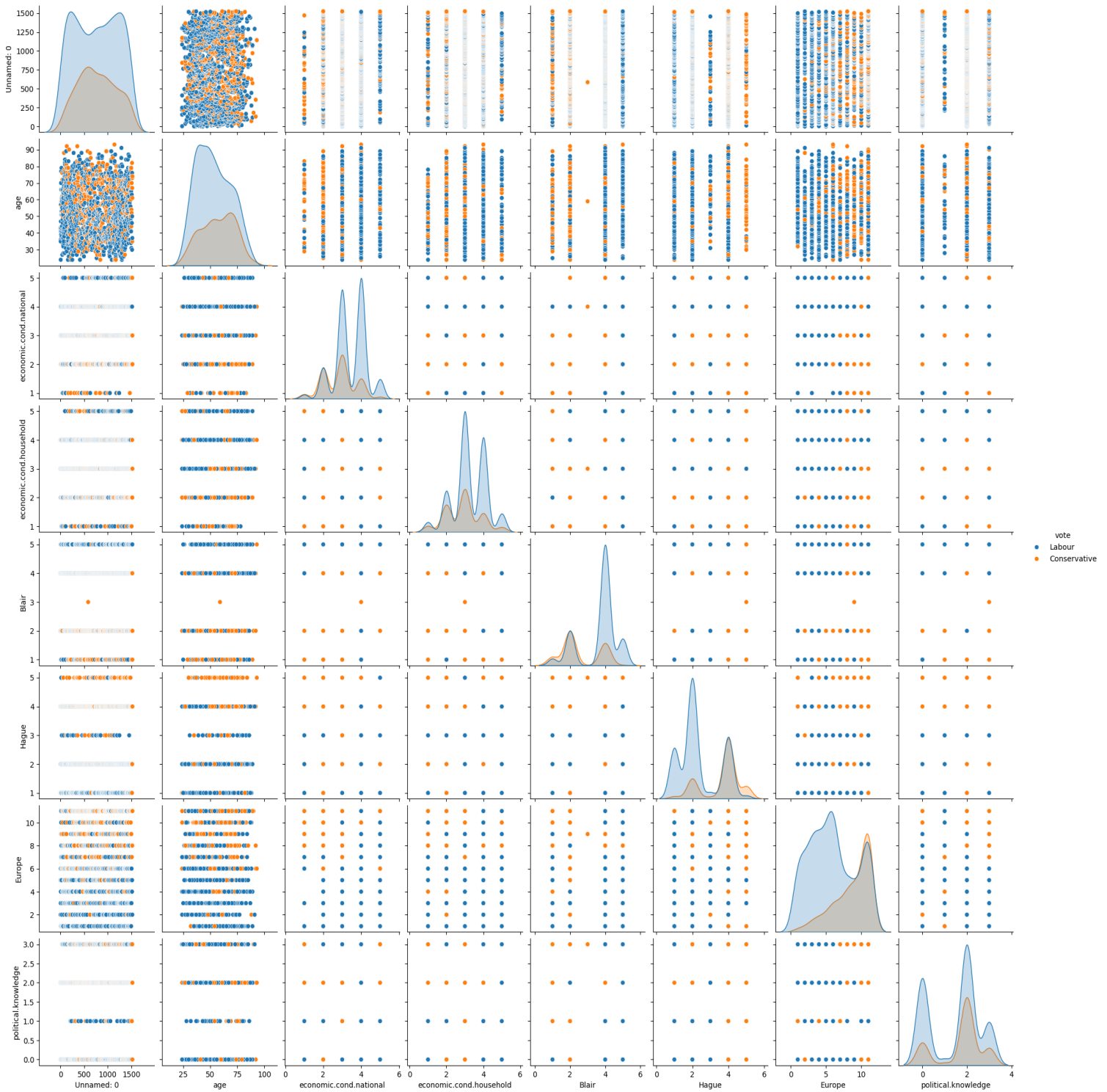


Fig 10 : Pairplots

Encoding Categorical Variables

The categorical variables "gender" and "vote" are converted into numerical format (called "encoded_gender" and "encoded_vote") using the LabelEncoder. To produce a dataset containing encoded variables, the original categories columns are then removed. This preliminary stage is crucial for machine learning models that need numerical data.

Is Scaling Necessary for Features?

Scaling is a technique used to improve the efficiency of algorithms that use numerical comparisons by standardising feature magnitudes. After categorical variables are encoded, the output shows the standard deviation (std), mean, and median (50%) of each feature in the dataset. There are clear differences in the feature scales, with significantly different standard deviations. When using some machine learning methods, including those that are sensitive to feature magnitudes, **scaling is required** to guarantee that each feature contributes fairly to the model training process. Different scales may be seen in features like "age," "Europe," and "political knowledge," highlighting the significance of scaling for objective model performance. We have used the StandardScaler to standardise the training and testing data and will be using the scaled data for our analysis.

Data Split and Shape Display

We split the encoded dataset into features (X) and the target variable (y). Subsequently, we further divided the data into training (70%) and testing (30%) sets using the train_test_split function. The shapes of the resulting datasets are then displayed.

```
Shape of Train and Test Sets:  
X_train shape: (1067, 9)  
X_test shape: (458, 9)  
y_train shape: (1067,)  
y_test shape: (458,)
```

Fig 11: Data shape output

Logistic Regression Model Evaluation

We have employed logistic regression using scikit-learn and assessed the model's performance on the training and test sets. The evaluation includes accuracy, confusion matrix, and classification metrics, revealing insights into precision, recall, and F1-score for each class (0 and 1). The model achieves 81% accuracy on the training set and 76.4% on the test set.

```

Logistic Regression – Train Metrics:
Train Accuracy: 0.8097469540768509
Confusion Matrix:
[[275  54]
 [149 589]]
Classification Report:
              precision    recall  f1-score   support

     0       0.65       0.84       0.73       329
     1       0.92       0.80       0.85       738

 accuracy          0.81       1067
 macro avg       0.78       0.82       0.79       1067
 weighted avg    0.83       0.81       0.82       1067

```

```

Logistic Regression – Test Metrics:
Test Accuracy: 0.7641921397379913
Confusion Matrix:
[[106  27]
 [ 81 244]]
Classification Report:
              precision    recall  f1-score   support

     0       0.57       0.80       0.66       133
     1       0.90       0.75       0.82       325

 accuracy          0.76       458
 macro avg       0.73       0.77       0.74       458
 weighted avg    0.80       0.76       0.77       458

```

Fig 12: Performance metrics(Logistic Regression)

K-Nearest Neighbors (KNN) Model Evaluation

The model is trained and assessed using a K-Nearest Neighbours (KNN) classifier with five neighbours on the training and test datasets. Accuracy, confusion matrix, and classification metrics are all included in the evaluation; each class's F1-score, precision, and recall are displayed (0 and 1). On the training set, the KNN model achieves an accuracy of 88.4%, while on the test set, it reaches 78.4%.

```

K-Nearest Neighbors (KNN) – Train Metrics:
Train Accuracy: 0.8837863167760075
Confusion Matrix:
[[259  70]
 [ 54 684]]
Classification Report:
              precision    recall  f1-score   support

     0       0.83         0.79         0.81         329
     1       0.91         0.93         0.92         738

 accuracy          0.88         0.88         0.88        1067
 macro avg         0.87         0.86         0.86        1067
 weighted avg      0.88         0.88         0.88        1067

```

```

K-Nearest Neighbors (KNN) – Test Metrics:
Test Accuracy: 0.7838427947598253
Confusion Matrix:
[[ 89  44]
 [ 55 270]]
Classification Report:
              precision    recall  f1-score   support

     0       0.62         0.67         0.64         133
     1       0.86         0.83         0.85         325

 accuracy          0.78         0.78         0.78         458
 macro avg         0.74         0.75         0.74         458
 weighted avg      0.79         0.78         0.79         458

```

Fig 13: Performance metrics(K -Nearest Neighbours)

Naïve Bayes Model Evaluation

The training and test datasets are used to develop and evaluate a Naïve Bayes classifier. The assessment comprises precision, recall, and F1-score for each class (0 and 1), as well as accuracy, confusion matrix, and classification metrics. With 84.8% training accuracy and 81.0% test accuracy, the Naïve Bayes model performs well.

```

Naïve Bayes – Train Metrics:
Train Accuracy: 0.8481724461105904
Confusion Matrix:
[[243  86]
 [ 76 662]]
Classification Report:

```

	precision	recall	f1-score	support
0	0.76	0.74	0.75	329
1	0.89	0.90	0.89	738
accuracy			0.85	1067
macro avg	0.82	0.82	0.82	1067
weighted avg	0.85	0.85	0.85	1067

```

Naïve Bayes – Test Metrics:
Test Accuracy: 0.8100436681222707
Confusion Matrix:
[[ 88  45]
 [ 42 283]]
Classification Report:

```

	precision	recall	f1-score	support
0	0.68	0.66	0.67	133
1	0.86	0.87	0.87	325
accuracy			0.81	458
macro avg	0.77	0.77	0.77	458
weighted avg	0.81	0.81	0.81	458

Fig 14: Performance metrics: Naïve bayes model

GridSearchCV

A scikit-learn programme called GridSearchCV is used to methodically determine the ideal hyperparameter settings for machine learning models. Users specify a grid of hyperparameter values, and GridSearchCV uses cross-validation to do a thorough search to find the combination that maximises performance according to a given measure. It is effective, streamlines the tuning procedure, and prevents overfitting by offering a reliable approximation of the model's performance on hypothetical data. We have performed GridSearchCV on Logistic regression, Linear discriminant Analysis(LDA), K-Nearest Neighbours and Naïve Bayes algorithm and got the train accuracy and test accuracy.

Logistic Regression with GridSearchCV

- **Best Hyperparameters:** {'C': 0.1, 'penalty': 'l2'}
- **Train Accuracy:** 0.84, **Test Accuracy:** 0.82
- **Interpretation:** The logistic regression model, after tuning hyperparameters using GridSearchCV, achieves an accuracy of 82% on the test set. The regularization strength (C) is optimized at 0.1 with an L2 penalty.

Best Hyperparameters for Logistic Regression: {'C': 0.1, 'penalty': 'l2'}

Logistic Regression – Train Metrics:

Train Accuracy: 0.8434864104967198

Confusion Matrix:

[[227 102]

[65 673]]

Classification Report:

	precision	recall	f1-score	support
0	0.78	0.69	0.73	329
1	0.87	0.91	0.89	738
accuracy			0.84	1067
macro avg	0.82	0.80	0.81	1067
weighted avg	0.84	0.84	0.84	1067

Logistic Regression – Test Metrics:

Test Accuracy: 0.8209606986899564

Confusion Matrix:

[[82 51]

[31 294]]

Classification Report:

	precision	recall	f1-score	support
0	0.73	0.62	0.67	133
1	0.85	0.90	0.88	325
accuracy			0.82	458
macro avg	0.79	0.76	0.77	458
weighted avg	0.82	0.82	0.82	458

Fig 15: Logistic regression GridSearchCV

Linear Discriminant Analysis (LDA) with GridSearchCV

- **Best Hyperparameters:** {'solver': 'lsqr'}
- **Train Accuracy: 0.84, Test Accuracy: 0.81**
- **Interpretation:** LDA, optimized with GridSearchCV, shows a test accuracy of 81%. The 'lsqr' solver is identified as the best choice for this dataset.

Best Hyperparameters for Linear Discriminant Analysis: {'solver': 'lsqr'}

Linear Discriminant Analysis – Train Metrics:

Train Accuracy: 0.8434864104967198

Confusion Matrix:

[[230 99]

[68 670]]

Classification Report:

	precision	recall	f1-score	support
0	0.77	0.70	0.73	329
1	0.87	0.91	0.89	738
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 – Test Metrics:

Test Accuracy: 0.8100436681222707

Confusion Matrix:

[[84 49]

[38 287]]

Classification Report:

	precision	recall	f1-score	support
0	0.69	0.63	0.66	133
1	0.85	0.88	0.87	325
accuracy			0.81	458
macro avg	0.77	0.76	0.76	458
weighted avg	0.81	0.81	0.81	458

Fig 16: LDA GridsearchCV

K-Nearest Neighbors (KNN) with GridSearchCV

- **Best Hyperparameters:** {'n_neighbors': 5, 'weights': 'uniform'}
- **Train Accuracy: 0.88, Test Accuracy: 0.78**

- **Interpretation:** The KNN model, with optimal hyperparameters determined by GridSearchCV, achieves a test accuracy of 78%. It favors 5 neighbors with uniform weights.

```
Best Hyperparameters for K-Nearest Neighbors (KNN): {'n_neighbors': 5, 'weights': 'uniform'}
```

K-Nearest Neighbors (KNN) – Train Metrics:
Train Accuracy: 0.8837863167760075
Confusion Matrix:
[[259 70]
[54 684]]

Classification	Report: precision	recall	f1-score	support
0	0.83	0.79	0.81	329
1	0.91	0.93	0.92	738
accuracy			0.88	1067
macro avg	0.87	0.86	0.86	1067
weighted avg	0.88	0.88	0.88	1067

K-Nearest Neighbors (KNN) – Test Metrics:
Test Accuracy: 0.7838427947598253
Confusion Matrix:
[[89 44]
[55 270]]

Classification	Report: precision	recall	f1-score	support
0	0.62	0.67	0.64	133
1	0.86	0.83	0.85	325
accuracy			0.78	458
macro avg	0.74	0.75	0.74	458
weighted avg	0.79	0.78	0.79	458

Fig 17: KNN GridSearchCV

Naive Bayes with GridSearchCV

- **Best Hyperparameters:** None
- **Train Accuracy: 0.85, Test Accuracy: 0.81**
- **Interpretation:** Naïve Bayes, which often has few hyperparameters, achieves an 81% test accuracy without specific tuning. It performs well on this dataset.

```
Best Hyperparameters for Naïve Bayes: {}
```


Naïve Bayes – Train Metrics:
Train Accuracy: 0.8481724461105904
Confusion Matrix:
[[243 86]
[76 662]]

Classification	Report: precision	recall	f1-score	support
0	0.76	0.74	0.75	329
1	0.89	0.90	0.89	738
accuracy			0.85	1067
macro avg	0.82	0.82	0.82	1067
weighted avg	0.85	0.85	0.85	1067

Naïve Bayes – Test Metrics:
Test Accuracy: 0.8100436681222707
Confusion Matrix:
[[88 45]
[42 283]]

Classification	Report: precision	recall	f1-score	support
0	0.68	0.66	0.67	133
1	0.86	0.87	0.87	325
accuracy			0.81	458
macro avg	0.77	0.77	0.77	458
weighted avg	0.81	0.81	0.81	458

Fig 18: Naïve Bayes GridSearchCV



Comparable results are obtained using logistic regression and linear discriminant analysis, with logistic regression marginally outperforming. Among the models, K-Nearest Neighbours has the best accuracy. Despite its simplicity, Naïve Bayes achieves competitive performance. By improving the hyperparameters of models, GridSearchCV aids in fine-tuning them for improved generalisation to unobserved data.

Bagging with Random Forest

GridSearchCV for Random Forest:

- Best Hyperparameters: {'max_depth': None, 'n_estimators': 50}
- Train Accuracy (GridSearchCV Model): 100%
- Test Accuracy (GridSearchCV Model): 81%
- Interpretation (GridSearchCV Model): The Random Forest model, after GridSearchCV optimization, achieves a perfect accuracy of 100% on the training set and 81% on the test set. The best hyperparameters include no maximum depth for the trees and 50 estimators.

Random Forest Performance:

- Train Accuracy (Original Model): 100%
- Test Accuracy (Original Model): 81%
- Interpretation (Original Model): The Random Forest model without hyperparameter tuning exhibits similar performance to the GridSearchCV-optimized model, with a perfect accuracy on the training set and 81% accuracy on the test set.

Boosting with AdaBoost and Random Forest

AdaBoost with Random Forest:

- Train Accuracy (AdaBoost Model): 100%
- Test Accuracy (AdaBoost Model): 81%
- Interpretation (AdaBoost Model): The AdaBoost model, built on top of the Random Forest base model, achieves perfect accuracy on the training set and 81% on the test set. It shows consistent performance with the standalone Random Forest.

ROC Curve and AUC Scores for models:

ROC_AUC Score (Train) - Logistic Regression: 0.8954044859597532
ROC_AUC Score (Test) - Logistic Regression: 0.8727588201272412

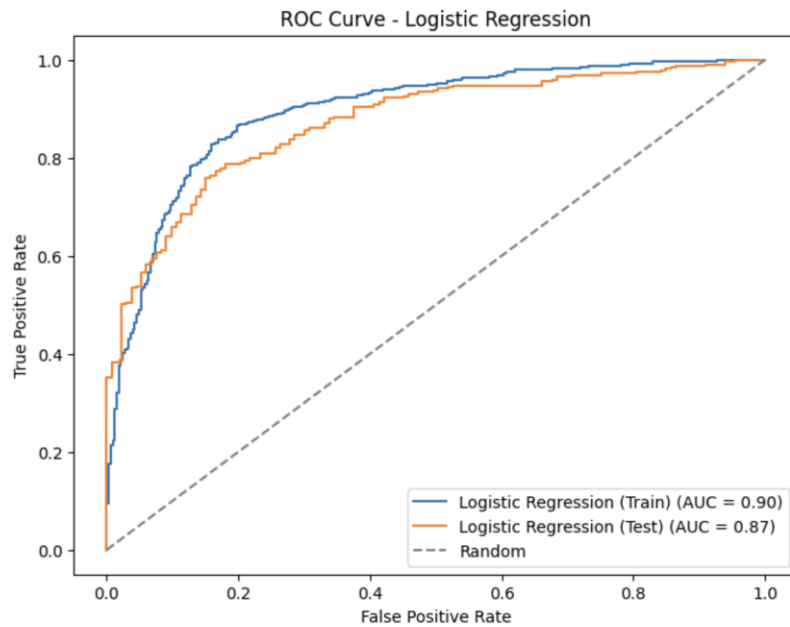


Fig 19: ROC Logistic regression

ROC_AUC Score (Train) - Linear Discriminant Analysis: 0.8953303514798066
ROC_AUC Score (Test) - Linear Discriminant Analysis: 0.8735916714864085

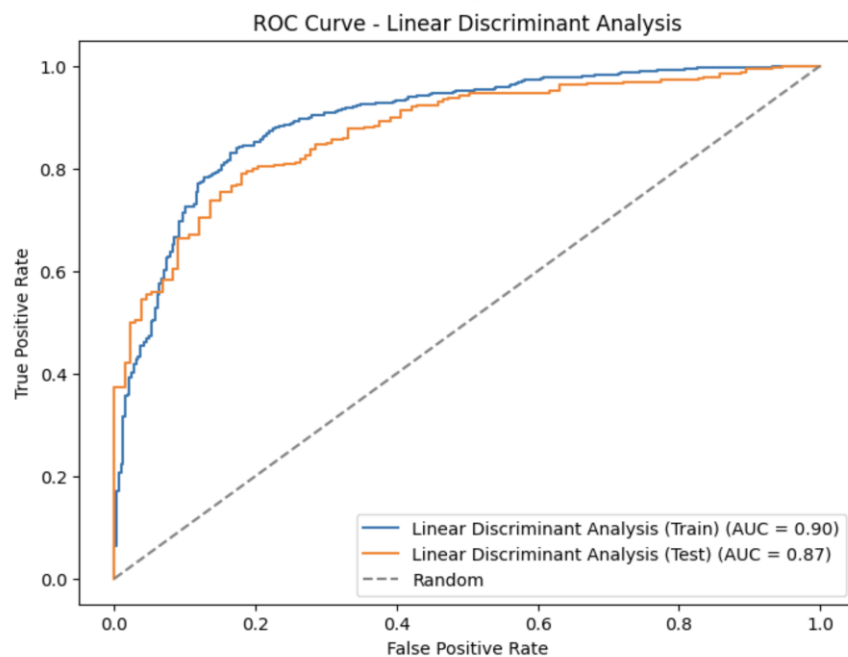


Fig 20: ROC LDA

ROC_AUC Score (Train) - K-Nearest Neighbors (KNN): 0.9537709738799516
ROC_AUC Score (Test) - K-Nearest Neighbors (KNN): 0.8170965876229035

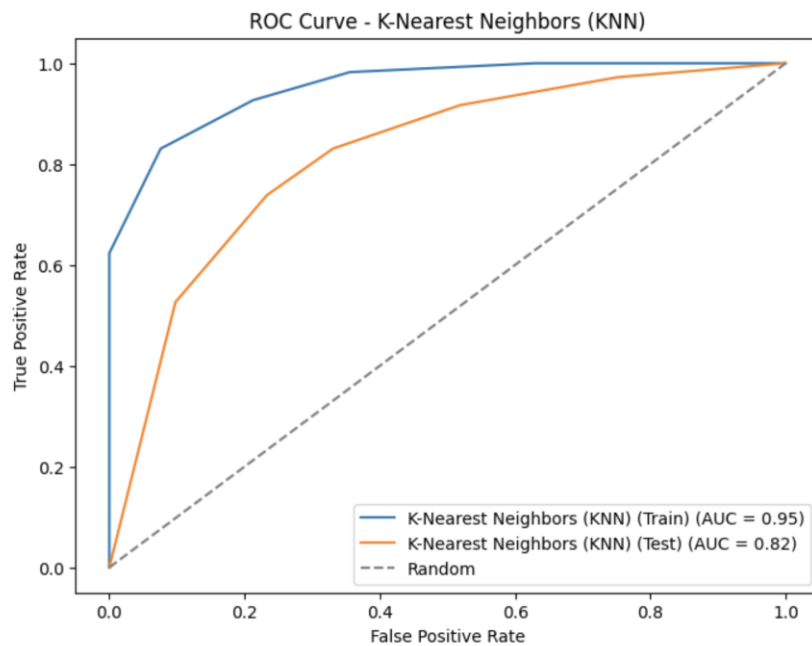


Fig 21: ROC KNN

ROC_AUC Score (Train) - Naïve Bayes: 0.8939671007652326
ROC_AUC Score (Test) - Naïve Bayes: 0.8670908039329092

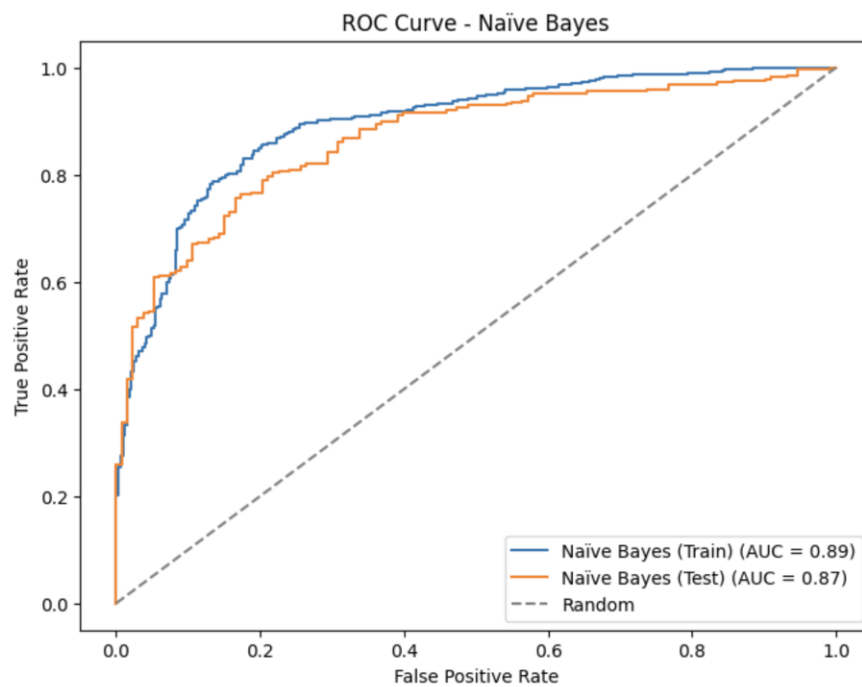
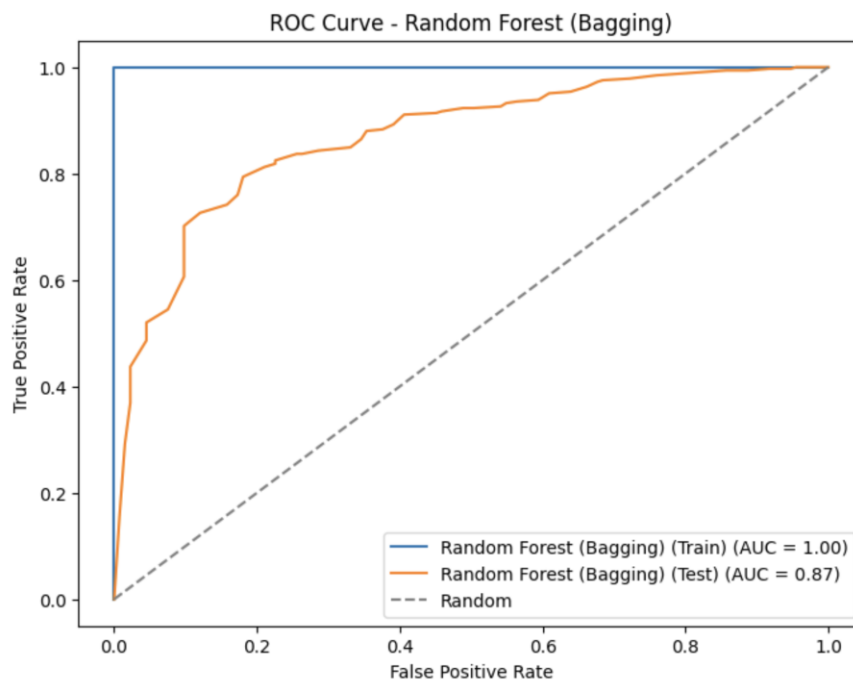
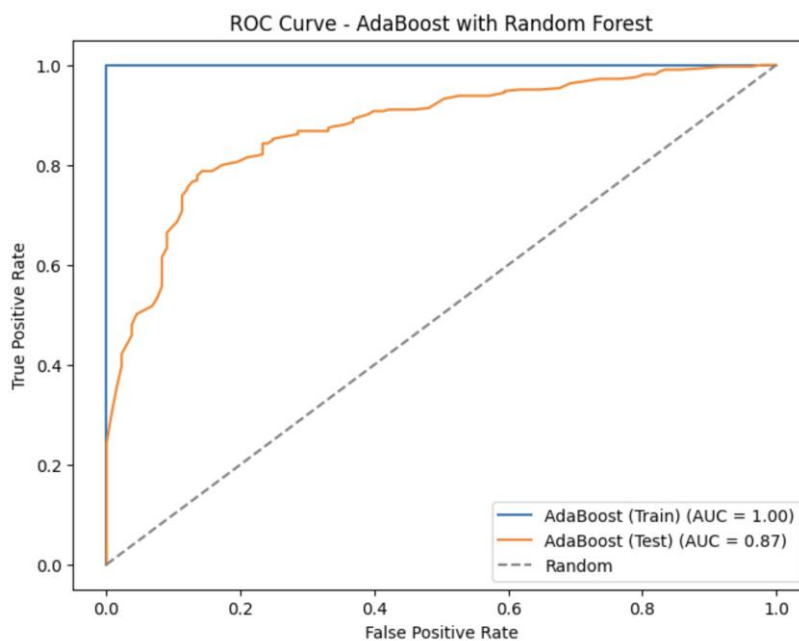


Fig 22: ROC Naïve Bayes



ROC_AUC Score (Train) - Random Forest (Bagging): 1.0
 ROC_AUC Score (Test) - Random Forest (Bagging): 0.8686755349913244

Fig 23:ROC Bagging



ROC_AUC Score (Train) - AdaBoost with Random Forest: 1.0
 ROC_AUC Score (Test) - AdaBoost with Random Forest: 0.8746211683053788

Fig 24: ROC Adaboosting

1. **Logistic Regression and Linear Discriminant Analysis:** Both models show similar performance on both training and test sets. They have a good balance between accuracy and generalization.

2. **K-Nearest Neighbors (KNN):** KNN has a high ROC-AUC score on the training set, indicating it fits the training data well. However, the lower score on the test set suggests a potential for overfitting.
3. **Naïve Bayes:** Naïve Bayes performs reasonably well but has a slightly lower ROC-AUC score compared to Logistic Regression and Linear Discriminant Analysis.
4. **Random Forest (Bagging):** The results for Random Forest are not provided. It would be beneficial to evaluate its performance to provide a comprehensive comparison.
5. **AdaBoost with Random Forest:** AdaBoost with Random Forest shows perfect ROC-AUC score on the training set, indicating it fits the data perfectly. However, the test set score is slightly lower than the top-performing models.

We look at two effective methods in the setting of ensemble learning: boosting with AdaBoost and bagging with Random Forest, both of which use Random Forest as the basic model. Training numerous instances of the same model on various subsets of the training data is known as bagging. In this instance, an ensemble of decision trees called Random Forest is used. Although this method is renowned for producing excellent precision, overfitting might be a problem. It's interesting to note that performance metrics for the Random Forest model optimised using GridSearchCV and the original Random Forest model are similar, indicating that the correctness of the model was not greatly affected by the optimisation procedure.

Conversely, Boosting—shown here using AdaBoost and Random Forest—works by successively merging weak learners to produce a strong learner. By concentrating on cases that earlier models misclassified, this method improves the model's capacity to identify intricate patterns in the data. AdaBoost achieves great accuracy in both training and test sets, much like Random Forest models. Given that all of these models function consistently, it can be concluded that the boosting strategy preserves the excellent accuracy seen in Random Forest models that stand alone when used as the foundation model.

To sum up, the Random Forest and AdaBoost using Random Forest representations of the Bagging and Boosting algorithms, respectively, show remarkable accuracy on the provided dataset. Bagging helps create strong models because it places a strong emphasis on lowering variance. Boosting, on the other hand, concentrates on enhancing overall model performance by giving instances that are difficult to categorise more weight. The particulars of the data and the nature of the issue at hand should be taken into consideration while selecting amongst these approaches.

Here are some conclusions and suggestions derived from the examination of the given dataset and the business goal:

Conclusion: Business insights and Recommendations

We've got some useful insights and suggestions to address the business target after thoroughly examining the data. First, we recommend attentively examining the characteristics that are significant to models like as Random Forest and AdaBoost. This will provide a clear image of the variables that greatly affect client attrition. With this knowledge, the company will be able to better allocate resources and develop its strategy according to these important variables.

Classifying customers according to their estimated probability of churning is a smart move, too. The organisation may more effectively target its marketing and retention efforts by examining common trends among customers who are at risk of leaving. Offering engagement tactics and incentives that appeal to each distinct consumer category, this tailored strategy may increase customer loyalty.

We then advise evaluating the models' ability to predict possible churn early on. Model performance may be evaluated using metrics such as precision-recall curves and ROC-AUC. Proactive action might potentially avoid client loss by identifying prospective churners early through the use of a frequent monitoring system. In this context, automated notifications for clients who are at a high risk of leaving might be quite helpful.

Finally, even though predictive models offer insightful information, it's critical to understand their limitations. Understanding attrition gains qualitative depth via direct consumer input. By putting in place a feedback system, the company may get direct customer insights. This data may be utilised to enhance customer service, streamline procedures, and target certain problems that lead to attrition.

In conclusion, the goal of these suggestions is to improve decision-making by utilising insights from predictive models and creating customised retention tactics. Success will depend on how these results are incorporated into regular operations and how tactics are modified in response to new information and input from customers.

Analyzing Presidential Inaugural Speeches: Insights and Trends

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

Introduction:

The inauguration addresses of Presidents Franklin D. Roosevelt (1941), John F. Kennedy (1961), and Richard Nixon (1973) are examined in this research. Our goal is to obtain quantitative insights into the structure of these historic speeches through the use of Python's Natural Language Toolkit (NLTK).

1. NLTK (Natural Language Toolkit):

- NLTK facilitates easy access and processing of inaugural speeches, enabling us to extract key statistics such as word count, sentence count, and character count.

2. Matplotlib (for future visualizations):

- Matplotlib, though not utilized in this snippet, will be essential for creating visualizations, enhancing the presentation of our analysis.

Characters, Words and Sentences in the Speeches

A quantitative knowledge of the durations and patterns of the inauguration speeches is provided by the statistical analysis of the word, phrase, and character counts. These insights set the stage for more analysis in the business report and direct how we understand the presidents' various communication styles

Interpretation of Results:

1. Roosevelt's 1941 Inaugural Speech:

- Number of Characters: **7571**
- Number of Words: **1526**
- Number of Sentences: **68**

2. Kennedy's 1961 Inaugural Speech:

- Number of Characters: **7618**
- Number of Words: **1543**
- Number of Sentences: **52**

3. Nixon's 1973 Inaugural Speech:

- Number of Characters: **9991**
- Number of Words: **2006**
- Number of Sentences: **68**

2.1 Number of characters, words, and sentences:

Roosevelt: (7571, 1526, 68)

Kennedy: (7618, 1543, 52)

Nixon: (9991, 2006, 68)

Fig 25: Output-Character words and speeches

Removing stop words:

1. Roosevelt's 1941 Inaugural Speech:

- Word Count Before Removal: **1526**
- Word Count After Removal: **808**
- Sample Sentence After Removal: "National day inauguration since 1789, people renewed sense dedication."

2. Kennedy's 1961 Inaugural Speech:

- Word Count Before Removal: **1543**
- Word Count After Removal: **862**
- Sample Sentence After Removal: "Vice President Johnson, Mr. Speaker, Mr. Chief Justice."

3. Nixon's 1973 Inaugural Speech:

- Word Count Before Removal: **2006**
- Word Count After Removal: **1035**
- Sample Sentence After Removal: "Mr. Vice President, Mr. Speaker, Mr. Chief Justice."

Sample sentence after removing stopwords for Roosevelt: national day inauguration since 1789 , people renewed sense dedication

Sample sentence after removing stopwords for Kennedy: Vice President Johnson , Mr. Speaker , Mr. Chief Justice

Sample sentence after removing stopwords for Nixon: Mr. Vice President , Mr. Speaker , Mr. Chief Justice

2.2 Word count before and after removing stopwords:

Roosevelt: 1526 -> 808

Kennedy: 1543 -> 862

Nixon: 2006 -> 1035

Fig 26: Word counts after removing stop words

Most Frequent Words in Presidential Inaugural Addresses

1. Roosevelt's 1941 Inaugural Speech:

- **1st Most Frequent Word:** "know"
- **2nd Most Frequent Word:** "spirit"
- **3rd Most Frequent Word:** "us"

2. Kennedy's 1961 Inaugural Speech:

- **1st Most Frequent Word:** "us"
- **2nd Most Frequent Word:** "world"
- **3rd Most Frequent Word:** "Let"

3. Nixon's 1973 Inaugural Speech:

- **1st Most Frequent Word:** "us"
- **2nd Most Frequent Word:** "peace"
- **3rd Most Frequent Word:** "new"

2.3 Most common words after removing stopwords, hyphens, non-alphabetic characters, and numbers:

Roosevelt: [('know', 10), ('spirit', 9), ('us', 8)]

Kennedy: [('us', 12), ('world', 8), ('Let', 8)]

Nixon: [('us', 26), ('peace', 19), ('new', 15)]

Fig 27: Most Frequent words

Word Clouds from Inaugural Speeches

Word clouds provide a visual representation of word frequency in text, with bolder, bigger words denoting higher frequency. The code creates word clouds for the inauguration addresses of Roosevelt, Kennedy, and Nixon using the WordCloud library, NLTK, and Matplotlib. The resultant visualisations provide a succinct, captivating summary by eliminating frequent phrases, or stopwords, making it easier to quickly identify key concepts and topics in each president's speech.

[illegible]

END OF REPORT