greatlearning
Learning for Life

MODEL SOLUTION

Machine Learning Assessment
DSBA

Table of Contents

Problem - 1

Problem Statement

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

Dataset for Problem: Election Data

Data Dictionary for Election Data:

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

A. Data Ingestion

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

Load the required packages, set the working directory and load the data file.

We find one redundant column 'Unnamed: 0' which serves as an index. We shall drop the column as pandas automatically assign an index to the dataset.

Dataset has 1525 rows and 9 features (after removing the redundant index).

Let us start the data exploration step with the head method to look at the first five rows.

	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

Table1: DataFrame head with top five rows

A quick look at the dataset and data dictionary tells us that the independent variables consist of six ordinal categorical variables, one nominal categorical variable and One Continuous (integer) variable. Also all categorical variables except 'gender' are encoded (i.e. have numerical values). The target variable is a Binary categorical variable (nominal).

On checking for any duplicate values, we find that we have eight duplicated rows. We remove them using the drop_duplicates( ) method of the Data Frame. So our number of rows is reduced to 1517 (1525 – 8= 1517).
Let us now use the info ( ) and describe ( ) methods of the Data Frame –

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1517 entries, 0 to 1524
Data columns (total 9 columns):
vote                    1517 non-null object
age                     1517 non-null int64
economic.cond.national  1517 non-null int64
economic.cond.household 1517 non-null int64
Blair                   1517 non-null int64
Hague                   1517 non-null int64
Europe                  1517 non-null int64
political.knowledge     1517 non-null int64
gender                  1517 non-null object
dtypes: int64(7), object(2)
memory usage: 118.5+ KB
```

Table 2: Dataset information

To utilize the full potential of the describe ( ) method, we temporarily convert the encoded categorical columns into object type (in place of int64) so that they are treated as categorical variables and then use the method. We get the following results –

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
vote	1517	2	Labour	1057	NaN	NaN	NaN	NaN	NaN	NaN	NaN
age	1517	NaN	NaN	NaN	54.2413	15.7017	24	41	53	67	93
economic.cond.national	1517	5	3	604	NaN	NaN	NaN	NaN	NaN	NaN	NaN
economic.cond.household	1517	5	3	645	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Blair	1517	5	4	833	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Hague	1517	5	2	617	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Europe	1517	11	11	338	NaN	NaN	NaN	NaN	NaN	NaN	NaN
political.knowledge	1517	4	2	776	NaN	NaN	NaN	NaN	NaN	NaN	NaN
gender	1517	2	female	808	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Table 3: Statistical description using the describe ( ) method

From table 3, we get information about the most frequently repeated values for the categorical variables. In addition, we find that the mean age is quite high (54.24 years), and the minimum age for voting seems to be 24 years (or the opinion of younger people is not considered in the study).

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

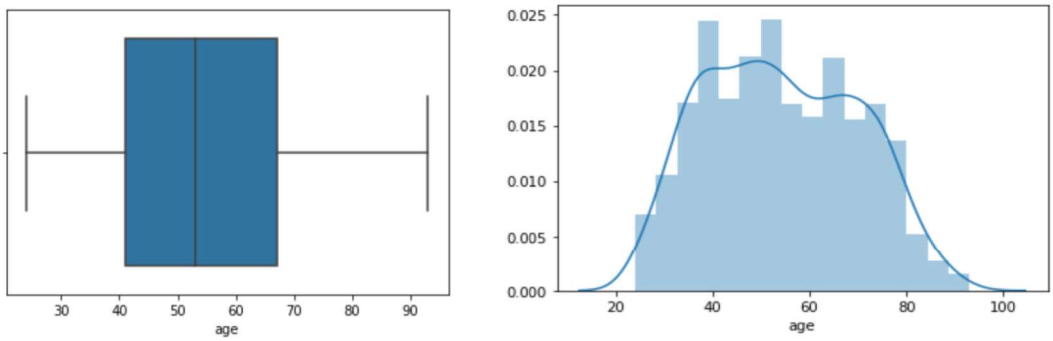
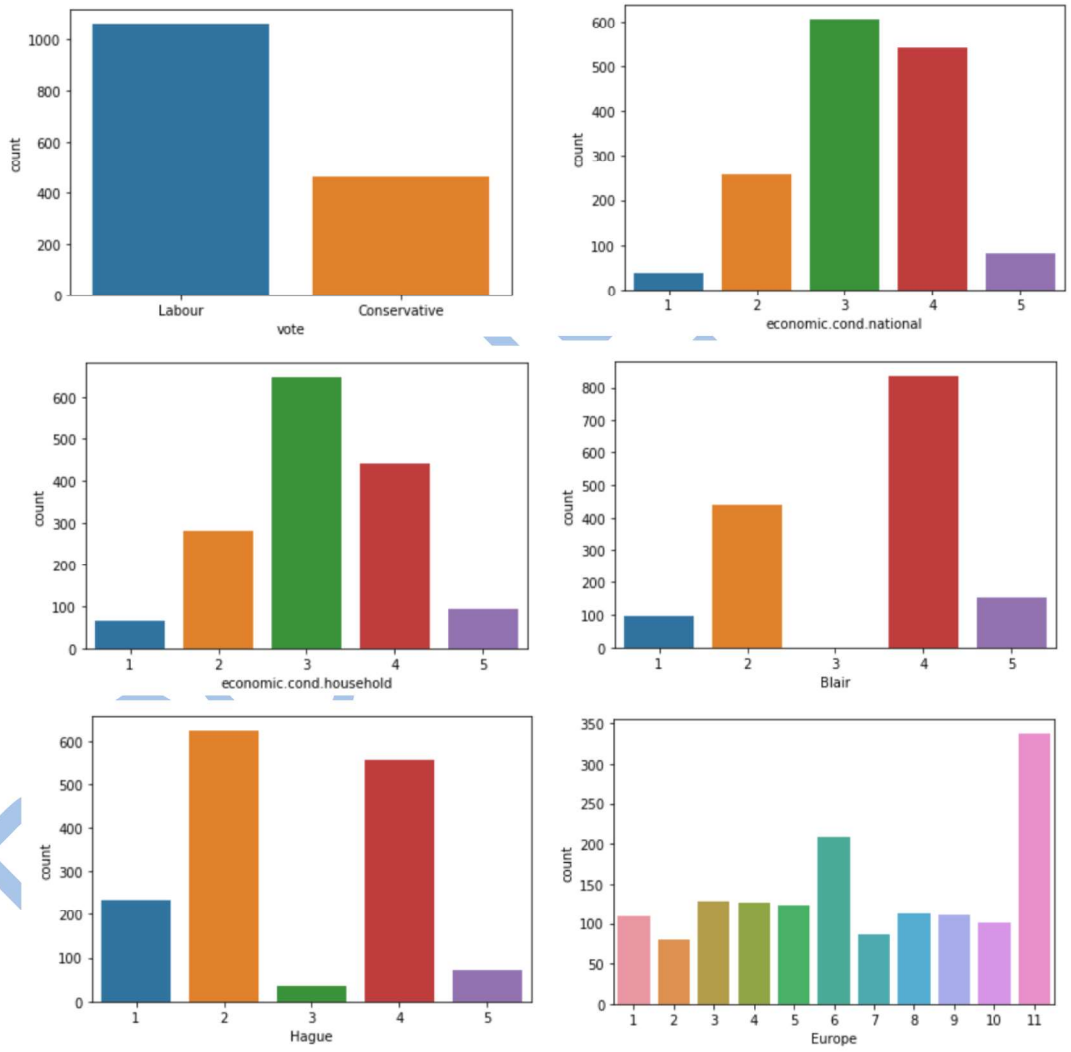


Figure 1: a) Box plot of variable 'age' b) Normalized histogram and KDE plot of variable 'age'



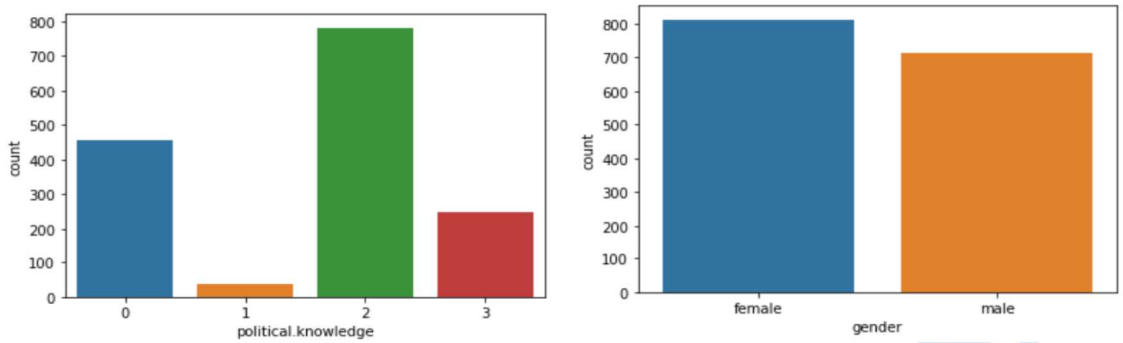


Figure 2: Count plots of the categorical variables

Inferences on Univariate analysis

1. No outliers present in 'age' variable. (As evident from the Box plot)
2. 'age' Does not follow Normal Distribution and has Kurotosis of -0.94 which implies too flat distribution. (k=0 for normal distribution)
3. Dependent variable 'vote' has a higher proportion of 'Labour' over 'Conservative'
4. Economic conditions of household and national level follow distribuiton as per common knowledge
5. The plots of 'Hague' and 'Blair' suggest that very few people have neutral opinions. 'Blair' has more positive opinions, whereas 'Hague' has more negative.
6. 'Europe' variable has good proportions of all the categories, only 11 being an exception
7. 'gender' variable has an almost equal proporion of male and female observations

Bivariate Analysis

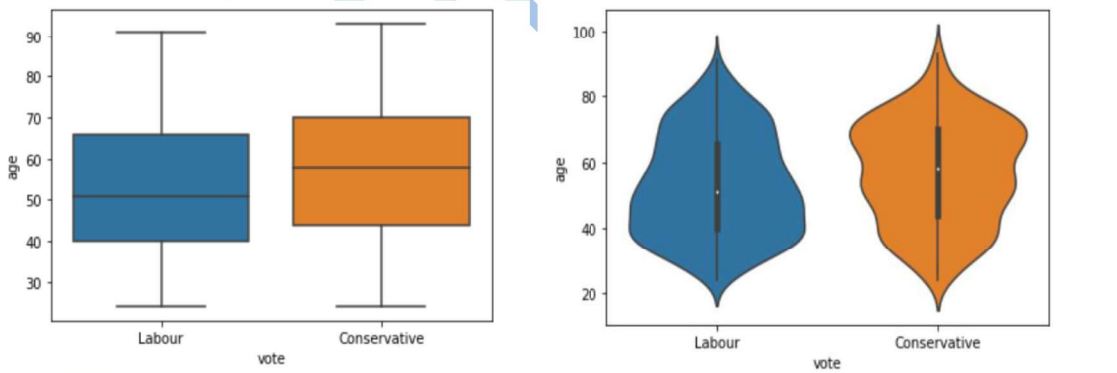


Figure 3 : a) 'age' vs 'vote' Box Plot *b) 'age' vs 'vote' Violin Plot*

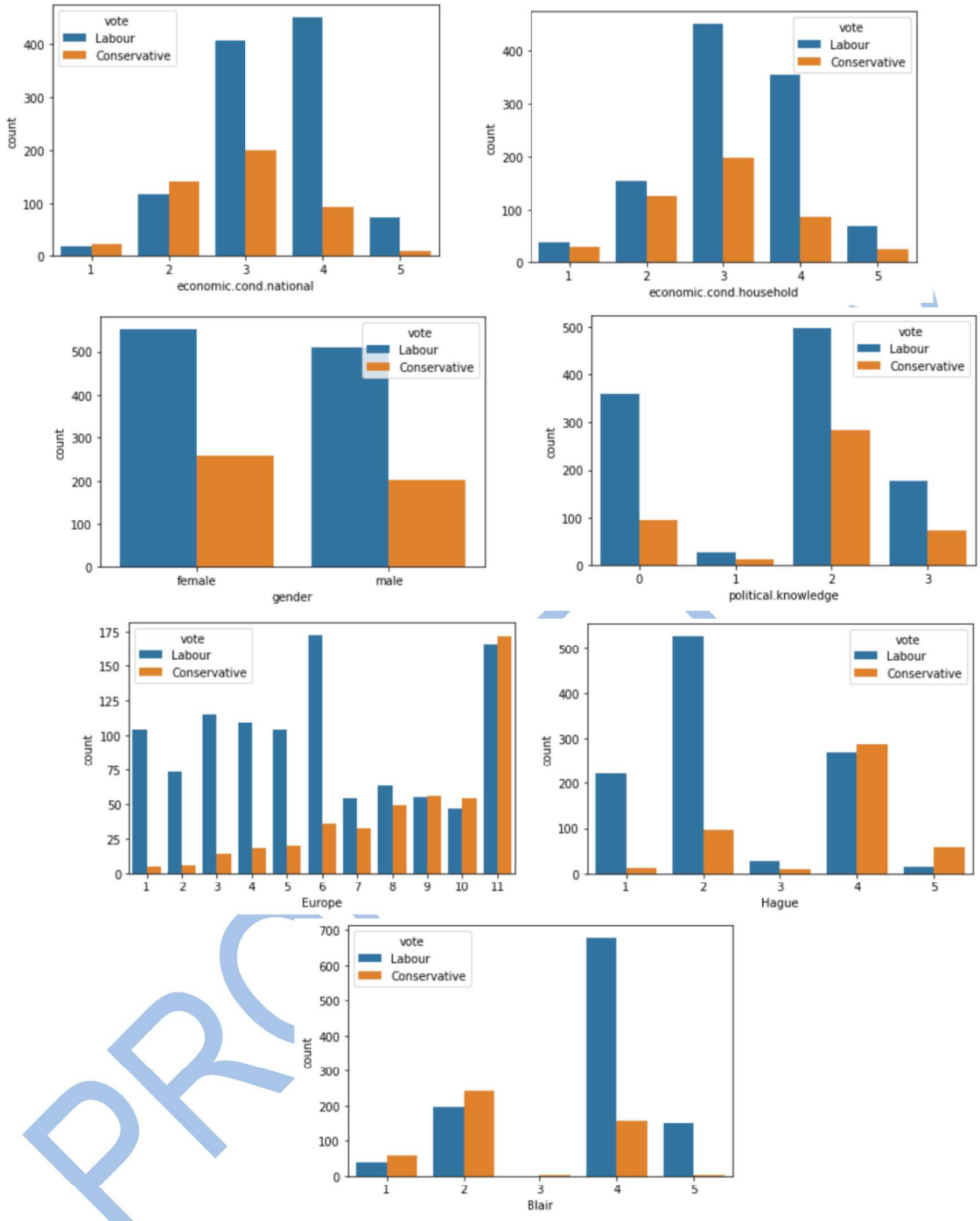


Figure 4: 'vote' vs Categorical variables Count Plots

Inferences from Bivariate Analysis

1. Elder people (60+) tend to vote for 'Conservative' part (As clear from the box plots and violin plot)
2. People who have lower assessments for economic conditions both national and household tend to vote for 'Conservative'

3. An almost equal number of voters across gender
4. People with stronger Eurosceptic sentiments tend to vote for 'Conservative'
5. Hague belongs to Conservaive party, whereas Blair is in Labour party

Data Preparation:

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

We encode the 'gender' variable by using pd.get_dummies( ) and dropping the first column to make sure we don't have a pair of colinear variables in our data set. We will also manually encode the 'vote' variable.

Whether Scaling is required or not - Scaling of variables is optional for linear models such as Linear regression, Logistic Regression, LDA and tree based models such as Random forest. However, scaling is a neccesity when using Distance based models such as KNN and SVM because we do not want variables with greater numerical values to be given more importance by the model.

It should be noted, that scaling should only be done after train test split, meaning the parameters(mean, or min-max) learned from the train set be applied to scale the test set to make sure that test data is unseen during the training phase.

As none of the features follows a Normal distribution, we can use Min-Max Scaler for our problem.

We now proceed with splitting the data set into train and test. We use the train_test_split function with arguments test_size = 0.3 to get 70:30 split , random_state = 100 for reproduceable results and stratify = y(target variable) so that target variable has equal proportion of classes in train and test set.

The train set now has 1061 rows, 8 features whereas the test set has 456 rows, 8 features.

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
958	54	2	2	2	2	9	2	1
980	56	2	2	4	1	6	2	1
882	69	3	3	2	2	11	2	0
665	80	5	3	2	5	9	0	1
637	56	3	3	2	4	3	2	0

Table 4: Train set top five rows

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
1521	73	2	2	4	4	8	2	1
96	41	3	3	4	4	5	2	0
39	72	1	3	2	2	11	2	0
1088	76	4	4	5	2	1	0	0
447	40	4	4	4	4	1	0	1

Table 5: Test set top five rows

Modelling:

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

The results after applying **Logistic Regression** on the train and test sets are as follows –

	precision	recall	f1-score	support
0	0.78	0.67	0.72	322
1	0.87	0.91	0.89	739
accuracy			0.84	1061
macro avg	0.82	0.79	0.81	1061
weighted avg	0.84	0.84	0.84	1061

Table 6: Classification report train set

```
Confusion matrix for train set :
 [[217 105]
 [ 63 676]]
```

Table 7: Confusion Matrix train set

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

	precision	recall	f1-score	support
0	0.72	0.69	0.70	138
1	0.87	0.88	0.88	318
accuracy			0.82	456
macro avg	0.79	0.79	0.79	456
weighted avg	0.82	0.82	0.82	456

Table 8: Classification report test set

```
Confusion matrix for test set :
 [[ 95  43]
 [ 37 281]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 9: Confusion Matrix test set

ROC-AUC Scores-

Train Set: 0.889 || Test Set: 0.884

In Logistic Regression, we can look at the feature coefficient values and deduce how much each feature influences the result/decision. Plotting the coefficient values in a Bar plot for better understanding.

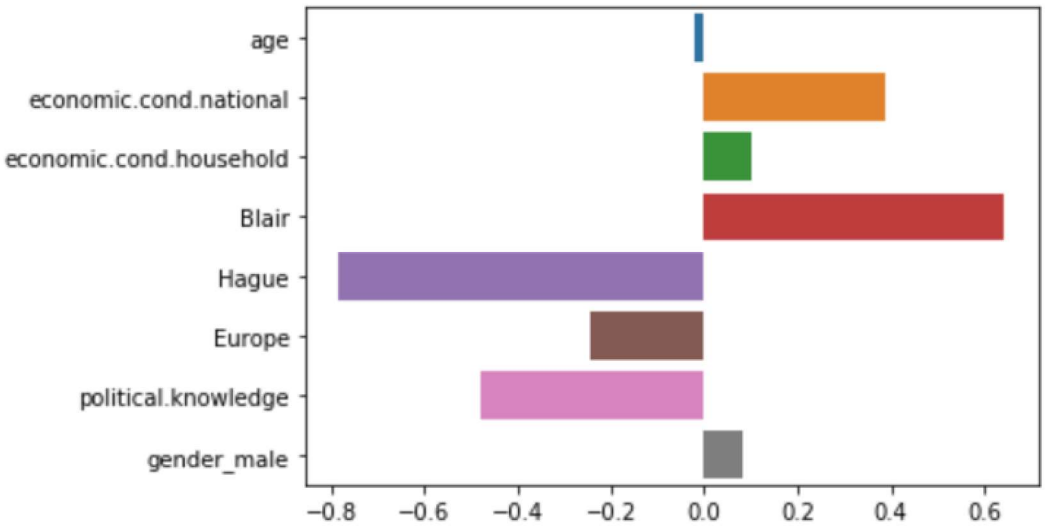


Figure 5: Bar plot of feature coefficeint values

Inferences-

1. Independent variables with larger absolute values have a greater impact on the target variable
2. Order of importance of variables - 'Hague'(negative influence) > 'Blair'(positive influence) > 'political.knowledge' (negative influence) >'economic.cond.national' (positive influence) > 'Europe' (negative influence)
3. Variables 'age', 'economic.cond.household' & 'gender_male' are least important
4. Train accuracy and test accuracy & ROC-AUC scores are very similar, hence this model does not suffer from overfitting

The results after applying **LDA** on train and test sets are as follows –

	precision	recall	f1-score	support
0	0.77	0.68	0.72	322
1	0.87	0.91	0.89	739
accuracy			0.84	1061
macro avg	0.82	0.80	0.81	1061
weighted avg	0.84	0.84	0.84	1061

Table 10: Classification report train set

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0	True Negative	False Positive

```
Confusion matrix for train set :
 [[220 102]
 [ 66 673]]
```

Conservative		
Actual Class 1 Labour	False Negative	True Positive

Table 11: Confusion Matrix train set

	precision	recall	f1-score	support
0	0.71	0.68	0.70	138
1	0.86	0.88	0.87	318
accuracy			0.82	456
macro avg	0.79	0.78	0.78	456
weighted avg	0.82	0.82	0.82	456

Table 12: Classification report test set

```
Confusion matrix for test set :
 [[ 94  44]
 [ 38 280]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 13: Confusion Matrix test set

ROC-AUC Scores-

Train Set: 0.888 || Test Set: 0.883

In the LDA model as well, we can look at the feature coefficient values and deduce how much each feature influences the result/decision. Plotting the coefficient values in a Bar plot for better understanding.

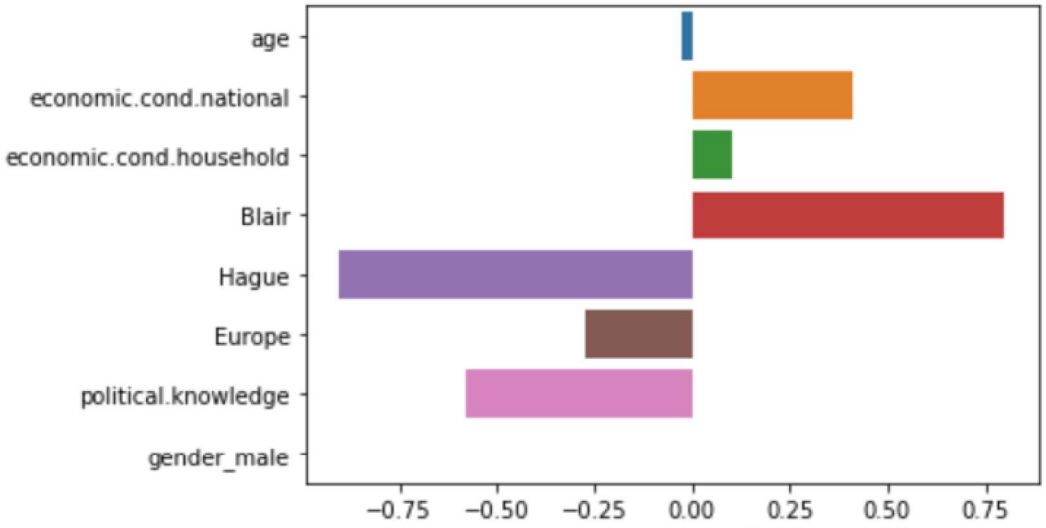


Figure 6: Bar plot of feature coefficeint values

Inferences -

1. Order of importance of predictors- 'Hague' > 'Blair' > 'political.knowlede' >'economic.cond.national' > 'Europe'
2. Predictors 'age', 'gender_male', 'economic.cond.household' are least important
3. Train accuracy and test accuracy & ROC-AUC scores are very similar, hence this model does not suffer from overfitting

2. Apply KNN Model, Naïve Bayes Model and support vector machine (SVM) model. Interpret the results.

KNN Model (Without tuning Hyperparameters) –

Before using the KNN model, we will scale the data sets using MinMax Scaler from sklearn library. Results of applying the un-tuned KNN model are as follows-

	precision	recall	f1-score	support
0	0.81	0.75	0.78	322
1	0.89	0.92	0.91	739
accuracy			0.87	1061
macro avg	0.85	0.83	0.84	1061
weighted avg	0.87	0.87	0.87	1061

Table 14: Classification report train set

```
Confusion matrix for train set :
 [[241  81]
 [ 58 681]]
```

Table 15: Confusion Matrix train set

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

	precision	recall	f1-score	support
0	0.67	0.72	0.69	138
1	0.87	0.85	0.86	318
accuracy			0.81	456
macro avg	0.77	0.78	0.78	456
weighted avg	0.81	0.81	0.81	456

Table 16: Classification report test set

```
Confusion matrix for test set :
 [[ 99  39]
 [ 49 269]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 17: Confusion Matrix test set

ROC-AUC Scores-

Train Set: 0.937 || Test Set: 0.849

KNN Model (After tuning Hyperparameters) –

The details of the range of Hyperparameters, and the best values are documented in Appendix.

The results of applying the tuned KNN model are as follows –

	precision	recall	f1-score	support
0	0.78	0.75	0.76	322
1	0.89	0.91	0.90	739
accuracy			0.86	1061
macro avg	0.83	0.83	0.83	1061
weighted avg	0.86	0.86	0.86	1061

Table 18: Classification report train set

```
Confusion matrix for train set :
 [[242  80]
 [ 70 669]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 19: Confusion Matrix train set

	precision	recall	f1-score	support
0	0.65	0.72	0.69	138
1	0.87	0.83	0.85	318
accuracy			0.80	456
macro avg	0.76	0.78	0.77	456
weighted avg	0.81	0.80	0.80	456

Table20: Classification report test set

```
Confusion matrix for test set :
 [[ 99  39]
 [ 53 265]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 21: Confusion Matrix test set

ROC-AUC Scores-

Train Set: 0.921 || Test Set: 0.870

Inferences -

1. A significant difference in ROC-AUC score of train and test set implies the untuned KNN model suffers from overfitting.
2. Hyperparameter tuning of select parameters reduces the difference in ROC-AUC score of train and test set making the model more generalised
3. The optimum number of neighbors come out to be 10 and the weights parameter 'uniform' gives better results

Naïve Bayes-

The results after applying Naïve Bayes Model on the train and test sets are as follows –

	precision	recall	f1-score	support
0	0.75	0.70	0.72	322
1	0.87	0.90	0.88	739
accuracy			0.84	1061
macro avg	0.81	0.80	0.80	1061
weighted avg	0.83	0.84	0.83	1061

Table22: Classification report train set

Predicted as Class 0 Conservative	Predicted as Class 1 Labour

```
Confusion matrix for train set :
 [[225  97]
 [ 77 662]]
```

Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 23: Confusion Matrix train set

	precision	recall	f1-score	support
0	0.70	0.72	0.71	138
1	0.88	0.86	0.87	318
accuracy			0.82	456
macro avg	0.79	0.79	0.79	456
weighted avg	0.82	0.82	0.82	456

Table 24: Classification report test set

```
Confusion matrix for test set :
 [[ 99  39]
 [ 43 275]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 24: Confusion Matrix test set

ROC-AUC Scores-

Train Set: 0.885 || Test Set: 0.880

Inferences-

1. The model is fairly generalised (have similar Accuracy scores and ROC AUC score for Train and Test set)

SVM Classifier (Without Hyperparameter tuning using linear kernel) –

The results after applying SVM Classifier on the train and test sets are as follows-

	precision	recall	f1-score	support
0	0.77	0.67	0.72	322
1	0.87	0.91	0.89	739
accuracy			0.84	1061
macro avg	0.82	0.79	0.80	1061
weighted avg	0.84	0.84	0.84	1061

Table 25: Classification report train set

```
Confusion matrix for train set :
 [[217 105]
 [ 65 674]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 26: Confusion Matrix train set

	precision	recall	f1-score	support
0	0.70	0.69	0.70	138
1	0.87	0.87	0.87	318
accuracy			0.82	456
macro avg	0.78	0.78	0.78	456
weighted avg	0.82	0.82	0.82	456

Table 27: Classification report test set

```
Confusion matrix for test set :
 [[ 95  43]
 [ 40 278]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 28: Confusion Matrix test set

ROC-AUC Scores-

Train Set: 0.888 || Test Set: 0.881

SVM Classifier (With Hyperparameter tuning) –

The details of the range of Hyperparameters, and the best values are documented in Appendix.

Results of applying the tuned SVM model are as follows –

	precision	recall	f1-score	support
0	0.81	0.71	0.76	322
1	0.88	0.93	0.90	739
accuracy			0.86	1061
macro avg	0.85	0.82	0.83	1061
weighted avg	0.86	0.86	0.86	1061

Table 29: Classification report train set

```
Confusion matrix for train set :
 [[229  93]
 [ 53 686]]
```

Table 30: Confusion Matrix train set

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

	precision	recall	f1-score	support
0	0.72	0.70	0.71	138
1	0.87	0.88	0.87	318
accuracy			0.82	456
macro avg	0.79	0.79	0.79	456
weighted avg	0.82	0.82	0.82	456

Table 31: Classification report test set

```
Confusion matrix for test set :
 [[ 96  42]
 [ 38 280]]
```

Table 32: Confusion Matrix test set

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

ROC-AUC Scores-

Train Set: 0.919 || Test Set: 0.887

Note- When using a linear kernel in SVM we can get the features coeffecient values. Let us visualize them using a Bar plot.

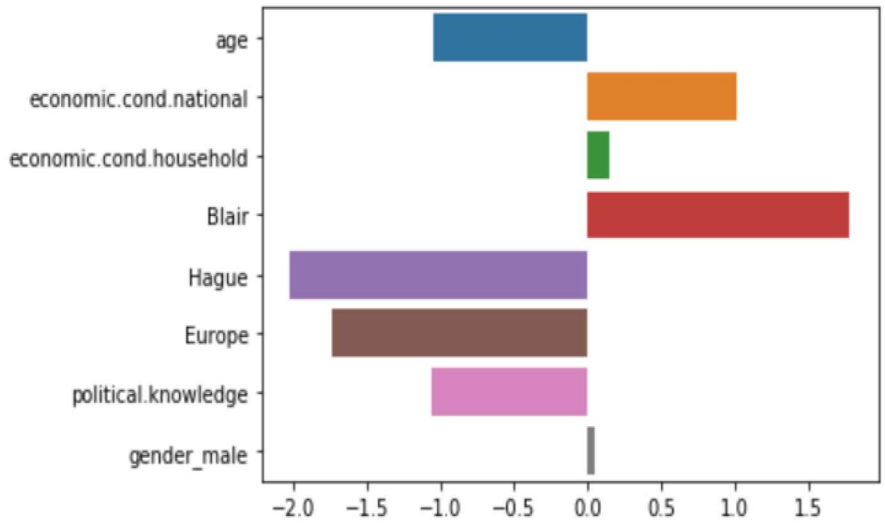


Figure 7: Bar plot of feature coeffecient values

Inferences-

1. Unlike previously trained models,' age' is a good predictor in the SVM model with a lienar kernel
2. 'gender_male' and 'economic.cond.household' are the two least important predictors
3. Untuned SVM with a lienar kernel and tuned SVM model do not suffer from overfitting
4. A slight increase in recall scores is observed after Hyperparamenter tuning

Bagging Classifier (Without Hyperparameter tuning)- **This part is not going to be considered for evaluation. This is done here only for extra knowledge**

Following are the results obtained after using un-tuned Bagging Classifier-

	precision	recall	f1-score	support
0	0.97	0.98	0.98	322
1	0.99	0.99	0.99	739
accuracy			0.98	1061
macro avg	0.98	0.98	0.98	1061
weighted avg	0.98	0.98	0.98	1061

Table 33: Classification report train set

```
Confusion matrix for train set :
 [[315   7]
 [  9 730]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 34: Confusion Matrix train set

	precision	recall	f1-score	support
0	0.66	0.67	0.66	138
1	0.85	0.85	0.85	318
accuracy			0.79	456
macro avg	0.76	0.76	0.76	456
weighted avg	0.79	0.79	0.79	456

Table 35: Classification report test set

```
Confusion matrix for test set :
 [[ 92  46]
 [ 48 270]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 36: Confusion Matrix test set

ROC-AUC Scores-

Train Set: 0.998 || Test Set: 0.843

Bagging Classifier (After Hyperparameter tuning)- **This part is not going to be considered for evaluation. This is done here only for extra knowledge**

The details of the range of Hyperparameters, and the best values are documented in Appendix. Following are the results obtained after using tuned Bagging Classifier -

	precision	recall	f1-score	support
0	0.89	0.49	0.63	322
1	0.81	0.97	0.89	739
accuracy			0.83	1061
macro avg	0.85	0.73	0.76	1061
weighted avg	0.84	0.83	0.81	1061

Table 37: Classification report train set

```
Confusion matrix for train set :
 [[157 165]
 [ 19 720]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 38: Confusion Matrix train set

	precision	recall	f1-score	support
0	0.75	0.46	0.57	138
1	0.80	0.93	0.86	318
accuracy			0.79	456
macro avg	0.77	0.70	0.71	456
weighted avg	0.78	0.79	0.77	456

Table 39: Classification report test set

```
Confusion matrix for test set :
 [[ 63  75]
 [ 21 297]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 40: Confusion Matrix test set

ROC-AUC Scores-

Train Set: 0.896 || Test Set: 0.881

Inferences -

1. Un-tuned Bagging classifier gets highly overfit as evident from the huge difference in the accuracy scores and ROC-AUC scores of train and test sets

2. Hyperparameter tuning did rectify the issue of overfit, but the tuned model suffers from underfitting evident from the poor recall and f1 scores.

Random Forest Classifier (Without Hyperparameter tuning)-

Following are the results obtained after using un-tuned Random Forest Classifier –

	precision	recall	f1-score	support
0	1.00	1.00	1.00	322
1	1.00	1.00	1.00	739
accuracy			1.00	1061
macro avg	1.00	1.00	1.00	1061
weighted avg	1.00	1.00	1.00	1061

Table 41: Classification report train set

```
Confusion matrix for train set :
 [[321   1]
 [  0 739]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 42: Confusion Matrix test set

	precision	recall	f1-score	support
0	0.70	0.69	0.70	138
1	0.87	0.87	0.87	318
accuracy			0.82	456
macro avg	0.78	0.78	0.78	456
weighted avg	0.82	0.82	0.82	456

Table 43: Classification report test set

```
Confusion matrix for test set :
 [[ 95  43]
 [ 40 278]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 44: Confusion Matrix test set

ROC-AUC Scores-

Train Set : 0.999 || Test Set : 0.880

Random Forest Classifier (After Hyperparameter tuning)-

The details of the range of Hyperparameters, and the best values are documented in Appendix.

Following are the results obtained after using tuned Random Forest Classifier –

	precision	recall	f1-score	support
0	0.88	0.71	0.79	322
1	0.88	0.96	0.92	739
accuracy			0.88	1061
macro avg	0.88	0.83	0.85	1061
weighted avg	0.88	0.88	0.88	1061

Table 45: Classification report train set

```
Confusion matrix for train set :
 [[229  93]
 [ 32 707]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 46: Confusion Matrix train set

	precision	recall	f1-score	support
0	0.74	0.67	0.70	138
1	0.86	0.90	0.88	318
accuracy			0.83	456
macro avg	0.80	0.79	0.79	456
weighted avg	0.83	0.83	0.83	456

Table 47: Classification report test set

```
Confusion matrix for test set :
 [[ 93  45]
 [ 33 285]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 48: Confusion Matrix test set

ROC-AUC Scores-

Train Set : 0.945 || Test Set : 0.894

In Random Forest models, we can take a look at the feature importance values. Plotting it on a Bar chart for better understanding –

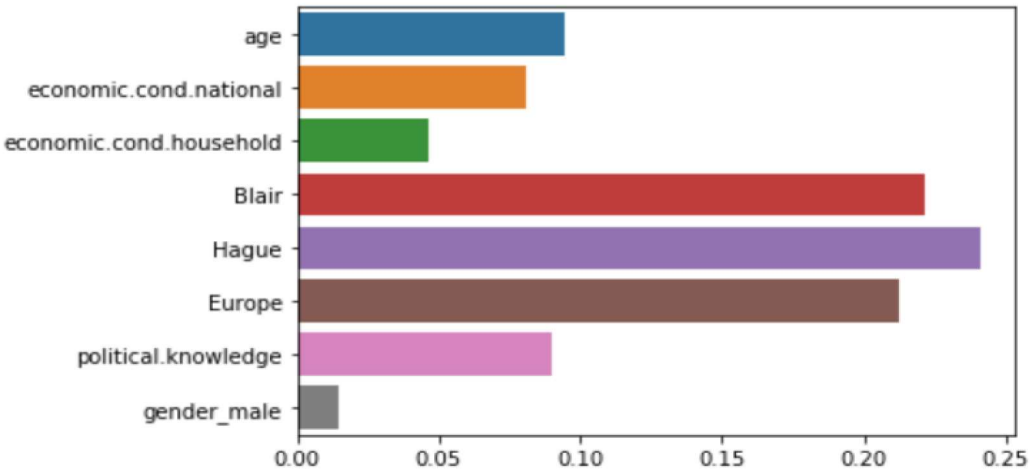


Figure 8: Bar chart of Feature importance

Inferences-

1. Un-tuned Random Forest models usually suffer from overfitting, just like we observed
2. After performing Hyperprameter tuning, a more genralised model was obtained. We were able to reduce the difference in Accuracy scores and ROC-AUC scores for train test sets.
3. 'age' turns out to be a fairly good feature in tuned Random Forest model
4. 'gender_male' and 'economic.cond.household' are the least important features

Boosting Classifier (Without Hyperparameter tuning) -

Following are the results obtained after using un-tuned Boosting Classifier –

	precision	recall	f1-score	support
0	0.87	0.77	0.82	322
1	0.91	0.95	0.93	739
accuracy			0.90	1061
macro avg	0.89	0.86	0.87	1061
weighted avg	0.90	0.90	0.90	1061

Table 49: Classification report train set

```
Confusion matrix for train set :
 [[249  73]
 [ 36 703]]
```

Table 50: Confusion Matrix train set

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

	precision	recall	f1-score	support
0	0.71	0.65	0.68	138
1	0.85	0.88	0.87	318
accuracy			0.81	456
macro avg	0.78	0.77	0.77	456
weighted avg	0.81	0.81	0.81	456

Table 51: Classification report test set

```
Confusion matrix for test set :
 [[ 90  48]
 [ 37 281]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 52: Confusion Matrix test set

ROC-AUC Scores-

Train Set : 0.953 || Test Set : 0.890

Boosting Classifier – (After Hyperparameter tuning)

The details of the range of Hyperparameters, and the best values are documented in Appendix. Following are the results obtained after using tuned Boosting Classifier –

	precision	recall	f1-score	support
0	0.93	0.83	0.88	322
1	0.93	0.97	0.95	739
accuracy			0.93	1061
macro avg	0.93	0.90	0.91	1061
weighted avg	0.93	0.93	0.93	1061

Table 53: Classification report train set

```
Confusion matrix for train set :
 [[268  54]
 [ 21 718]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 54: Confusion Matrix train set

	precision	recall	f1-score	support
0	0.69	0.64	0.66	138
1	0.85	0.87	0.86	318
accuracy			0.80	456
macro avg	0.77	0.76	0.76	456
weighted avg	0.80	0.80	0.80	456

Table 55: Classification report test set

```
Confusion matrix for test set :
 [[ 88  50]
 [ 40 278]]
```

	Predicted as Class 0 Conservative	Predicted as Class 1 Labour
Actual Class 0 Conservative	True Negative	False Positive
Actual Class 1 Labour	False Negative	True Positive

Table 56: Confusion Matrix test set

ROC-AUC Scores-

Train Set : 0.972 || Test Set : 0.873

In Boosting Classifier models, we can take a look at the feature importance values. Plotting it on a Bar chart for better understanding –

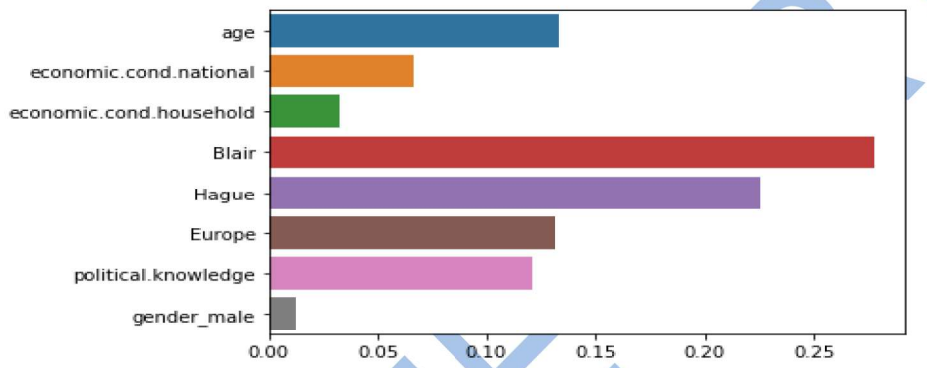


Figure 9: Bar plot showing feature importance values

Inferences-

1. Both un-tuned, as well as tuned Gradient Boosting models, suffer from a certain degree of overfitting
2. age again is a very good predictor
3. gender_male and economic.cond.household are the two least important predictors(features)

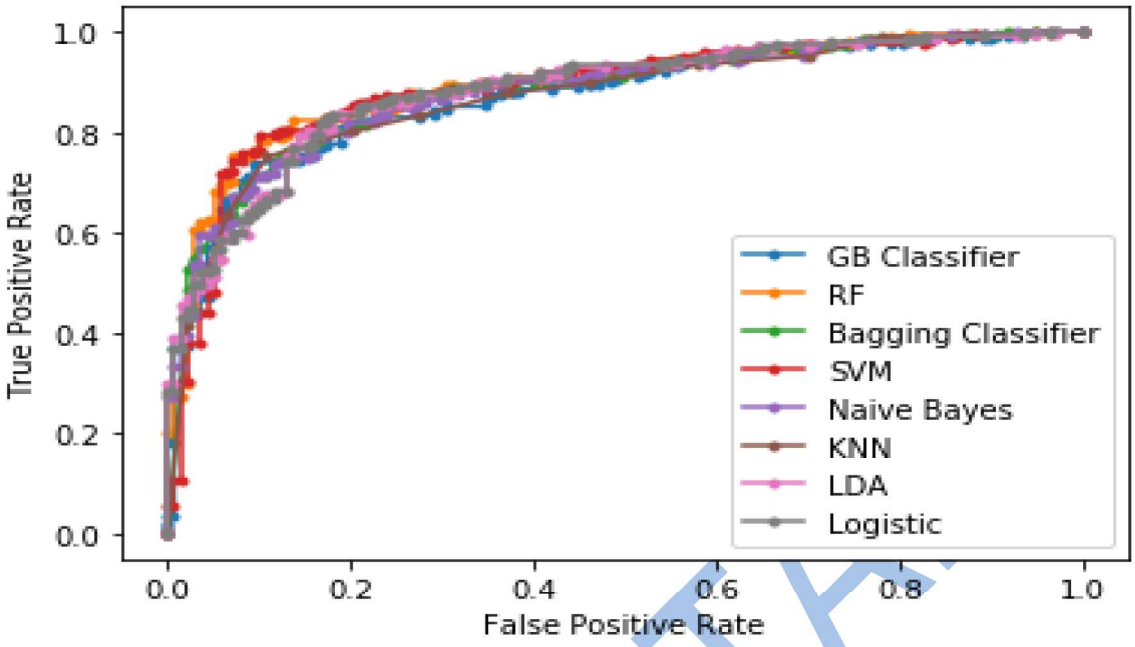


Figure 10: ROC- AUC curve for the models

Best Model

Model building is an iterative process. The model performance on both test and train dataset can be improved using feature engineering, feature extraction, hyperparameter tuning (by changing the combinations of various parameters). A model has to match the business objective and hence various permutation and combinations can be tried on to refine the model.

Below are the models that performed best on different metrics -

Accuracy scores - Tuned Random Forest Classifier
ROC AUC scores - Tuned Random Forest Classifier
Recall of minority class (Conservative) - Naïve Bayes Classifier
Recall of majority class (Labour) - Tuned Random Forest Classifier
f1 score of minority class (Conservative) - Naïve Bayes Classifier
f1 score of majority class (Labour) – Logistic Regression

Note – If two models perform equally well on an evaluation matric, then the simple one of the two should be selected.

Inference:

1. Based on these predictions, what are the insights?

Insights drawn from the analysis-

1. Bagging and Boosting models perform well provided the Hyperparameters are tuned properly(Time as well as a computationally expensive iterative task)
2. The model performance heavily depends on the type of input data and the distributions, for example in our dataset, even Linear models like Logistic regression performed fairly well.
3. Different models give a different level of importance to the input features. Hence, domain knowledge should be used to compliment the findings from models.

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

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

To find the number of characters we use the raw ( ) function, the number of words using the words( ) function and sentences using the sents( ) function of inaugural corpora.

Note- The words ( ) function also considers punctuation symbols as a word, hence we need to add a condition to exclude them from the count of words.

We make a DataFrame to store all the results. The contents of DataFrame are illusrated below-

	no_char	no_words	no_sents
1941-Roosevelt.txt	7571	1346	68
1961-Kennedy.txt	7618	1367	52
1973-Nixon.txt	9991	1816	69

Table 57: Result Dataset (post step 1)

2. Remove all the stopwords from all the three speeches.

Stopwords are the frequently occuring words that do not add value to the analysis, hence should be removed. NLTK package has an in-built list of 179 'stopwords'. We use this list to remove any occurrence of such words from the documents. Also, it is advised to convert all words to lower case, remove any numerical characters and punctuation marks from the documents as a part of the pre-processing step.

Sample text-

```
'Mr. Vice President, Mr. Speaker, Mr. Chief Justice, Senator Cook, Mrs. Eisenhower, and my fellow citizens of this great an
d good country we share together:\n\nWhen we met here four years ago, America wa'
```

Sample text after removal of stopwords -

```
' mr. vice president mr. speaker mr. chief justice senator cook mrs. eisenhower fellow citizens great good country share to
gether met four years ago america bleak spirit depressed prospect seemingly en'
```

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)

	no_char	no_words	no_sents	most_repeat	top_3
1941-Roosevelt.txt	7571	1346	68	know	['know', 'nation', 'us']
1961-Kennedy.txt	7618	1367	52	let	['let', 'us', 'sides']
1973-Nixon.txt	9991	1816	69	us	['us', 'let', 'new']

Table 58: Result dataset (post step 3)

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

Wordclouds can be plotted using the wordcloud python package. The resulting wordcloud for the three speeches are illustrated below –

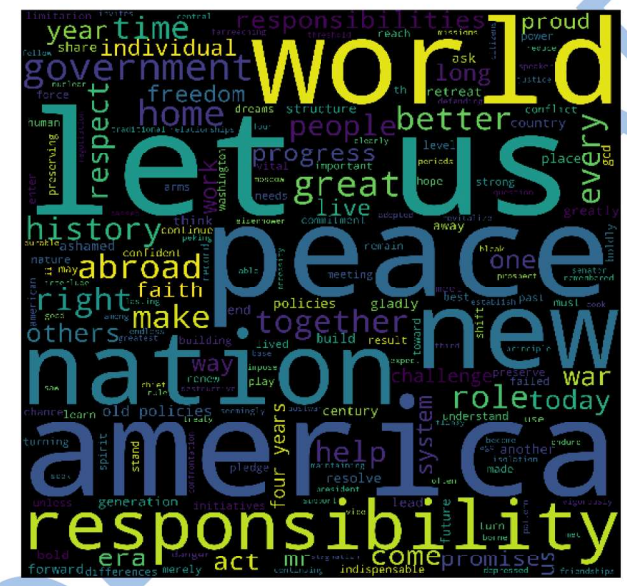


*Figure 10: Word Cloud for **President Richard Nixon's** inaugural address in 1973*

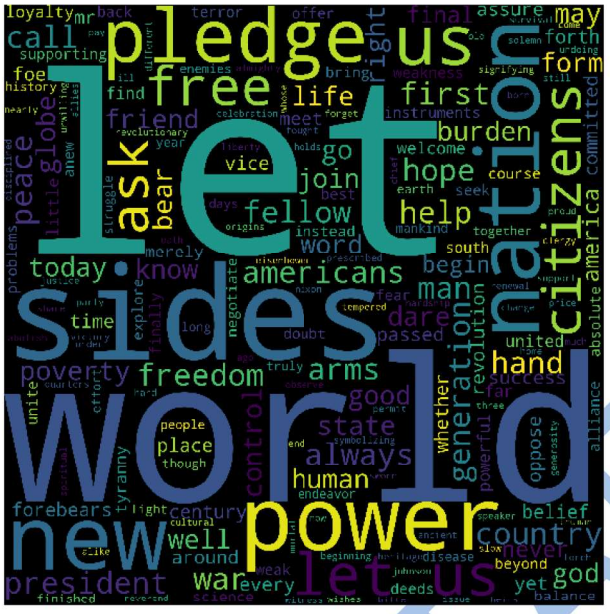


*Figure 11: Word Cloud for **President John F. Kennedy's** inaugural address in 1961*

*Figure 12: Word Cloud for **President Franklin D. Roosevelt's** inaugural address in 1941*

APPENDIX

Note- random_state = 100 was used in each Grid Search Object to get robust and reproducible results

1. KNN Hyperparameter tuning –

```
param_grid  = { 'n_neighbors' : [5,6,7,8,9,10],
                'weights':['uniform', 'distance'],
                'algorithm':['auto', 'ball_tree','kd_tree','brute'],
                'p': [1,2] }

best_params_ = {  'algorithm': 'auto',  'n_neighbors': 10,  'p': 2,
                  'weights': 'uniform' }
```

2. SVM Classifier Hyperparameter tuning-

```
param_grid   = { 'kernel' : [ 'linear' , 'rbf', 'poly'] ,
                'C' : np.logspace (-3, 2, 10) ,
                'gamma': [0.2,0.3,0.4,0.5,0.6],
                'degree':[2,3], 'random_state': [100] }

best_params_ = { 'C': 7.742636826811277,
                  'degree': 2,  'gamma': 0.3, 'kernel': 'rbf',
                  'random_state': 100 }
```

3. Bagging Classifier Hyperparameter tuning-

```
param_grid =  { 'base_estimator':[SVC( ), LogisticRegression( ) ,
                  DecisionTreeClassifier( )],
                  'max_samples':[0.5,0.6,0.7,0.8],
                 ' max_features':[0.3,0.4,0.5,0.6],
                  'n_estimators' :[30,40,50,100], 'random_state': [100] }

best_params_ = { 'base_estimator': SVC(C=1.0, degree=3,
                   gamma='scale', kernel='rbf' ), 'max_features': 0.5,
                   'max_samples': 0.6, 'n_estimators': 100,
                   'random_state': 100 }
```

4. Random Forest Hyperparameter tuning-

```
param_grid =  { 'max_depth':[5,6,7,8,9,10], 'min_samples_split':[8,9,10],
                 'n_estimators' :[100,200,300],'criterion': ["gini","entropy"],
                 ' random_state ' : [100] }

best_params_ = { 'criterion': 'entropy',  'max_depth': 6,
                   'min_samples_split': 10, 'n_estimators': 200,
                   'random_state': 100 }
```

5. Gradient Boosting Classifier Hyperparameter tuning-

```
param_grid = {'learning_rate': [0.05,0.075,0.1,0.2], 'max_depth':[5,6,7,8],
                'min_samples_split': [5,6,7,8,9,10 ] ,
                 'n_estimators':[50,75], 'random_state': [100] }

best_params_ = { 'learning_rate': 0.05, 'max_depth': 5,
                    'min_samples_split': 10, 'n_estimators': 75,
                  'random_state': 100 }
```