









MACHINE LEARNING PROJECT

Anay Tiwari

Case 1:

Build a machine learning model, to predict which party a voter will vote

Table of contents

-  **Project objective**
-  **Assumptions**
-  **Exploratory data analysis**
 -  **Summary of the dataset**
 -  **Bivariate analysis**
-  **Converting object data type into categorical**
-  **Splitting the data into train and test data**
 -  **Dimensions on the train and test data**

 **Model building**

 **Model Prediction**

 **Model evaluation**

 **Conclusion**

 **Recommendation**

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: 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.

Project Objective:

The Objective of the report is to explore the dataset "Election_Data.xlsx" in Python (JUPYTER NOTEBOOK) and generate insights about the dataset. This exploration report will consist of the following:

- Importing the dataset in jupyter notebook.
- Understanding the structure of dataset.
- Exploratory Data analysis
- Graphical exploration
- Prediction using various machine learning models
- Insights from the dataset

Assumptions:

Predictive modelling is the general concept of building a model that can make predictions. Typically, such a model includes a machine learning algorithm that learns certain properties from a training dataset to make those predictions. Pattern classification is to assign discrete class labels to observations as outcomes of a prediction.

Machine learning model predictions allow businesses to make highly accurate guesses as to the likely outcomes of a question based on historical data, which can be about all kinds of things. These provide the business with insights that result in tangible business value.

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

Dataset: Election_Data.xlsx

	Unnam ed: 0	vote	ag e	economic.cond.na tional	economic.cond.hous ehold	Blair	Hag ue	Euro pe	litical.know led	gend er
0	1	Labo ur	43	3	3	4	1	2	2	femal e
1	2	Labo ur	36	4	4	4	4	5	2	male
2	3	Labo ur	35	4	4	5	2	3	2	male
3	4	Labo ur	24	4	2	2	1	4	0	femal e
4	5	Labo ur	41	2	2	1	1	6	2	male

Information on dataset:

```
<class 'pandas.core.frame.DataFrame'> RangeIndex:
```

```
1525 entries, 0 to 1524
```

```
Data columns (total 9 columns):
```

#	Column	Non-Null Count	Dtype
0	vote	1525 non-null	object
1	age	1525 non-null	int64
2	economic.cond.national	1525 non-null	int64
3	economic.cond.household	1525 non-null	int64
4	Blair	1525 non-null	int64
5	Hague	1525 non-null	int64
6	Europe	1525 non-null	int64

```
7 political.knowledge      1525 non-null    int64    8    gender
   1525 non-null    object dtypes: int64(7), object(2) memory usage: 107.4+ KB
```

Inference

- The column “Unnamed : 0” is removed from the dataset before proceeding further as its insignificant for the analysis.
- There are 1525 rows and 9 columns
- Numerical Columns: age, economical_cond_national, economical_cond_household , Blair, Hague, Europe and political_knowledge.
- Non-Numerical Columns: vote and gender.
- There are no null values

Summary of the dataset:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
vote	1525	2	Labour	1063	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
age	1525	#N/A	#N/A	#N/A	54.1823	15.7112	24	41	53	67	93
economic.cond.national	1525	#N/A	#N/A	#N/A	3.2459	0.880969	1	3	3	4	5
economic.cond.household	1525	#N/A	#N/A	#N/A	3.14033	0.929951	1	3	3	4	5
Blair	1525	#N/A	#N/A	#N/A	3.33443	1.17482	1	2	4	4	5
Hague	1525	#N/A	#N/A	#N/A	2.74689	1.2307	1	2	2	4	5
Europe	1525	#N/A	#N/A	#N/A	6.72852	3.29754	1	4	6	10	11
political.knowledge	1525	#N/A	#N/A	#N/A	1.5423	1.08331	0	0	2	2	3
gender	1525	2	female	812	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A

Inference

- Vote and age variable have 2 unique values
- vote: Conservative and Labour
- Labour count is 1036 which is more than conservative count
- gender: Male and Female
- Female voters are more than male voters
- age is continuous variable
- minimum age of voters is 24
- maximum age of voters is 93
- Average age of voters is 54
- maximum Assessment of current national economic conditions is 5
- Average Assessment of current household economic conditions is 3
- minimum Assessment of the Labour leader is 1

○ Hague: average Assessment of the Conservative leader is 2.7 whereas Blair: average Assessment of the Labour leader is 3.3

Duplicates:

Number of duplicate rows = 8

Before (1525, 9)

After (1517, 9)

There are 8 duplicates in the dataframe which was dropped. now there are 1517 rows in the dataframe

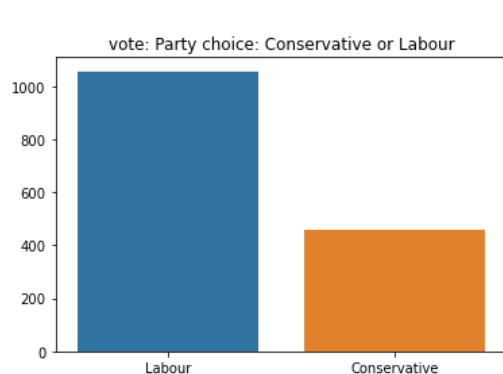
Skewness:

```
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 Inference

- Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean.
- age have positive skewness whereas other variables have negative skewness
- Blair has more skewness

Unique values for categorical variables vote:



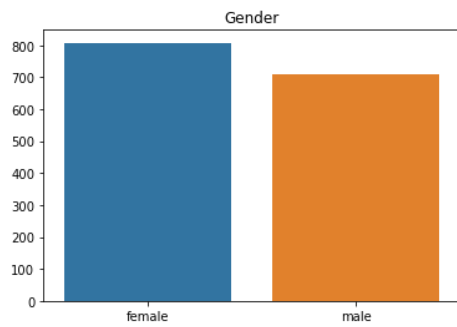
Party
choice:

Conservative or Labour

```
Labour          0.69677
Conservative     0.30323
Name: vote, dtype: float64
```

Inference

- nearly 69% vote for labour party only 30% vote for conservative party **Gender**



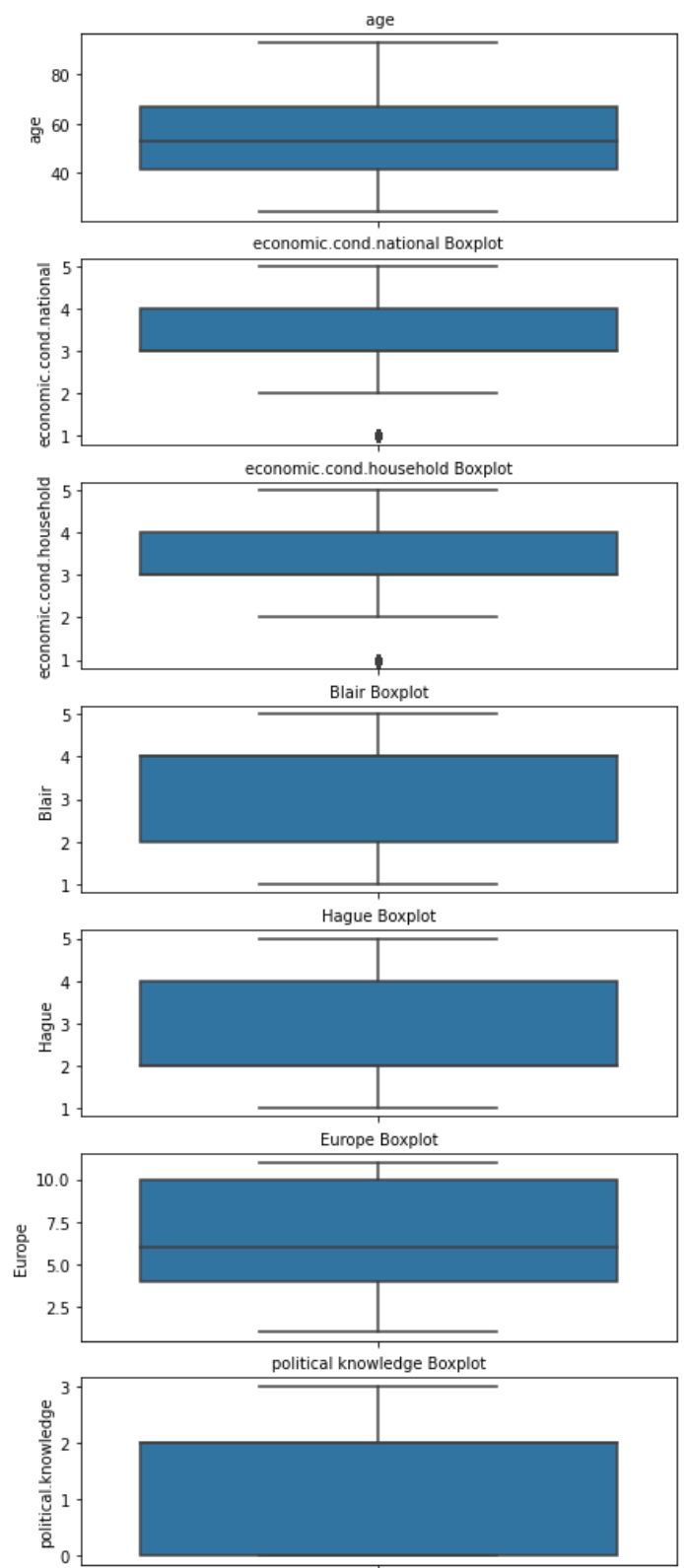
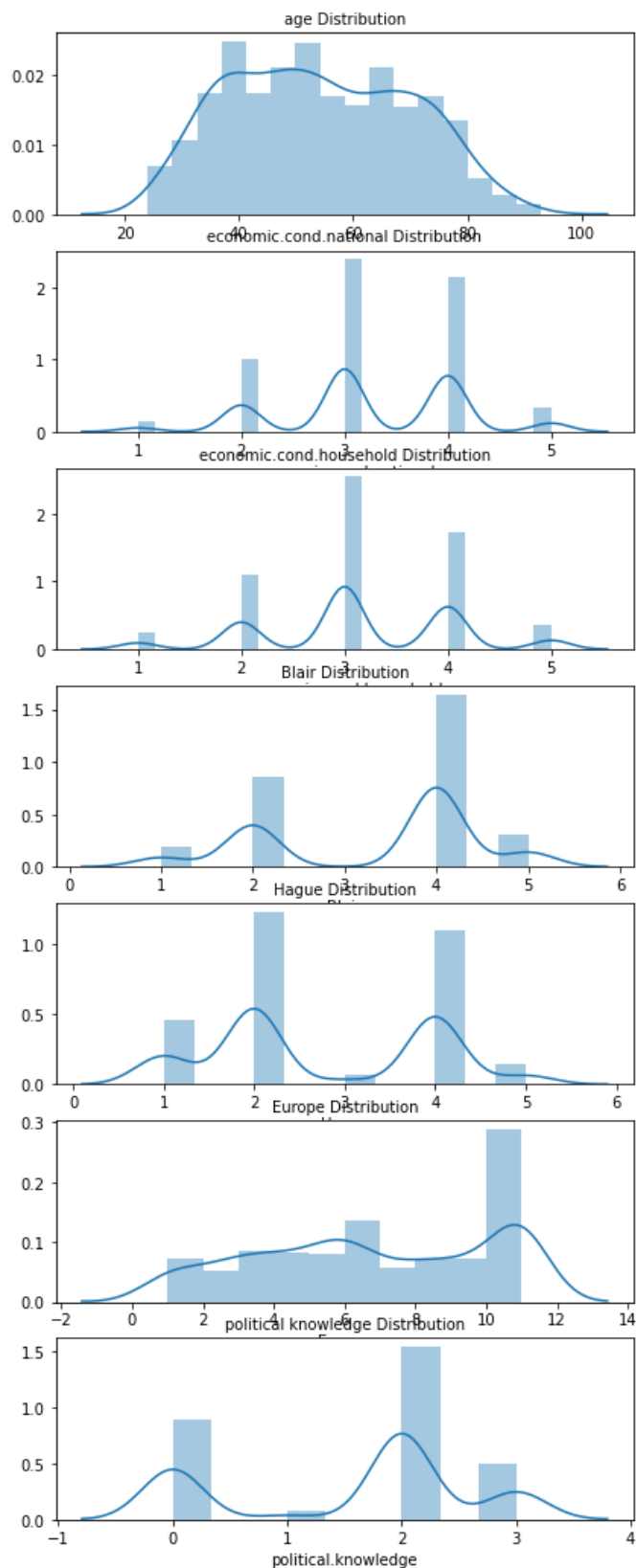
```
female    0.53263 male  
0.46737
```

```
Name: gender, dtype: float64
```

Inference

- 53% voters are female voters, and 46% voters are male voters

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

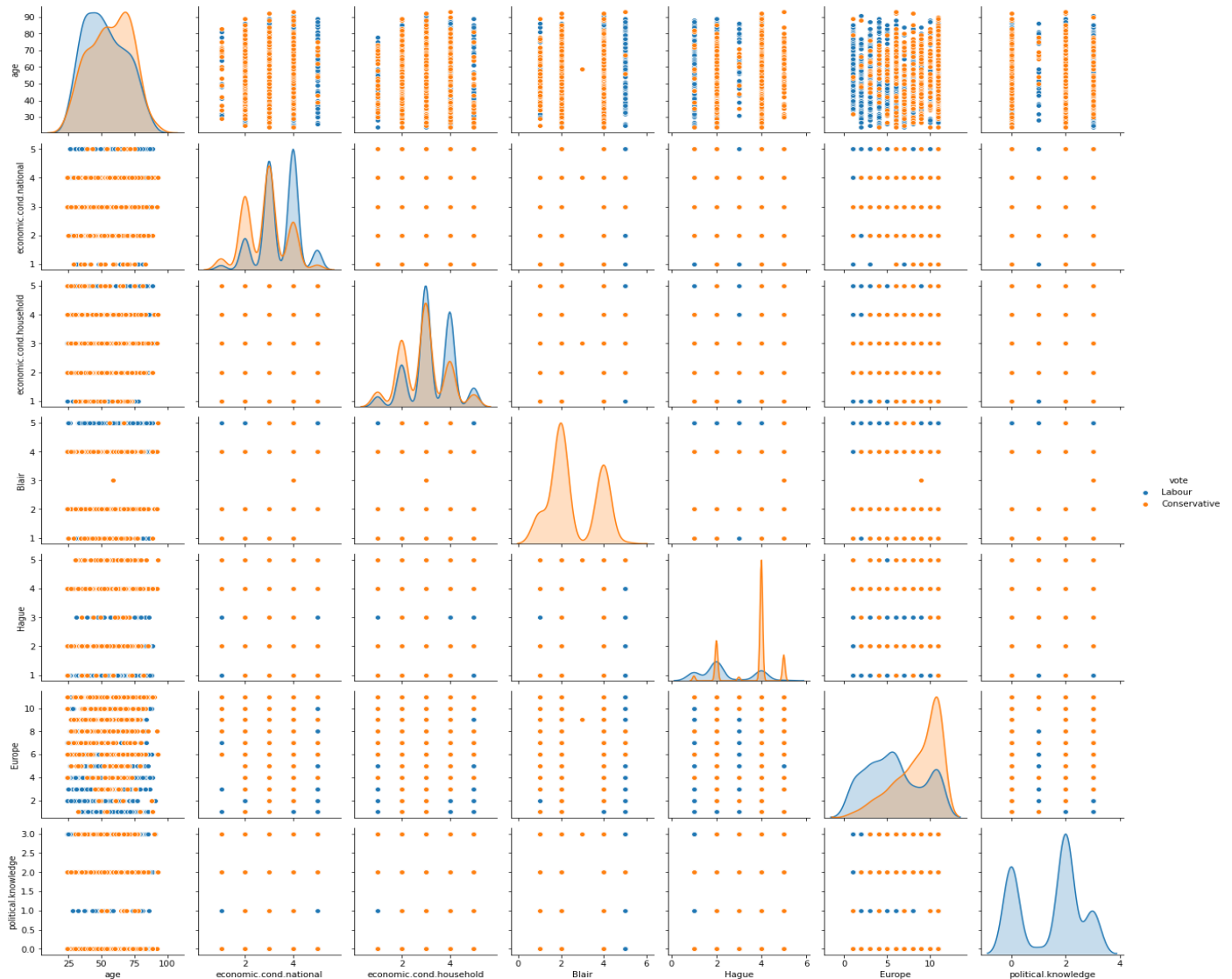


Inference

- only age variable is normally distributed and other variables has multimodal skewness seen
- only economic.cond.national and economic.cond.household have outliers

Bivariate Analysis:

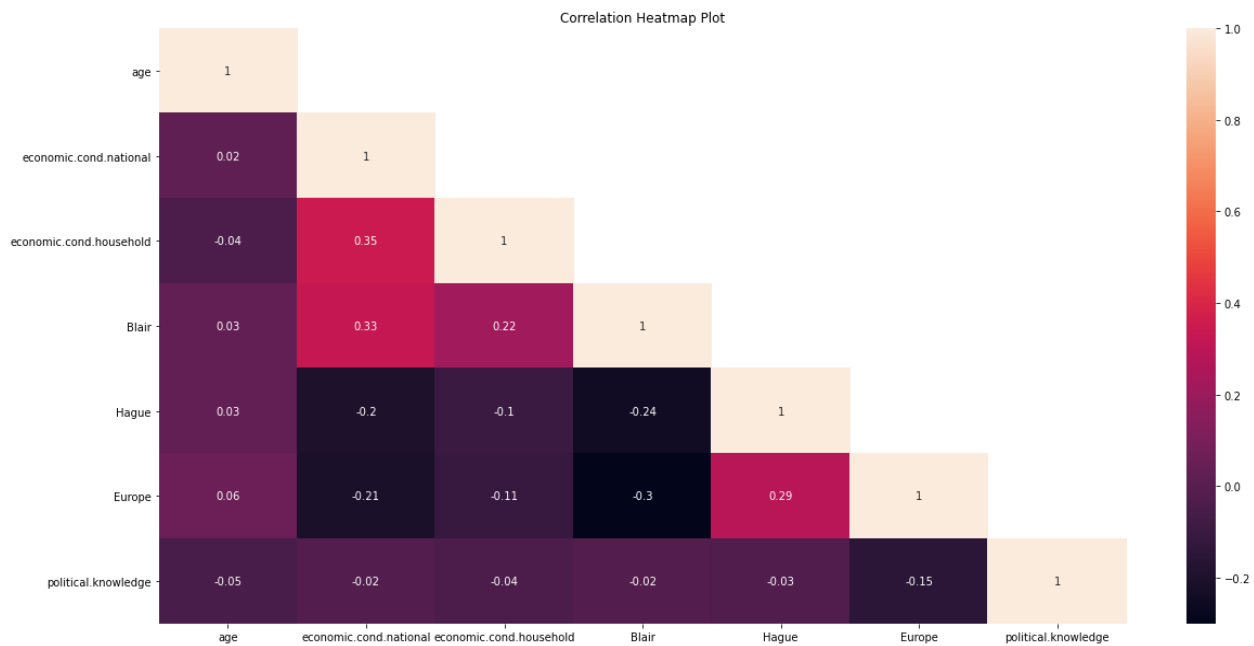
Pair plot between variables:



Inference

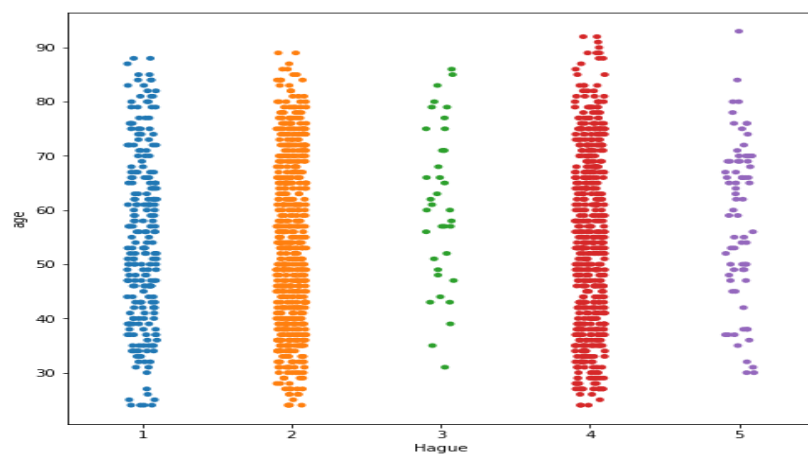
- There is no linear relationship between variables
- Some of the attributes look like they may have an exponential distribution
- Conservative party: Knowledge of parties' positions on European integration is unknown

Correlation Heatmap Plot



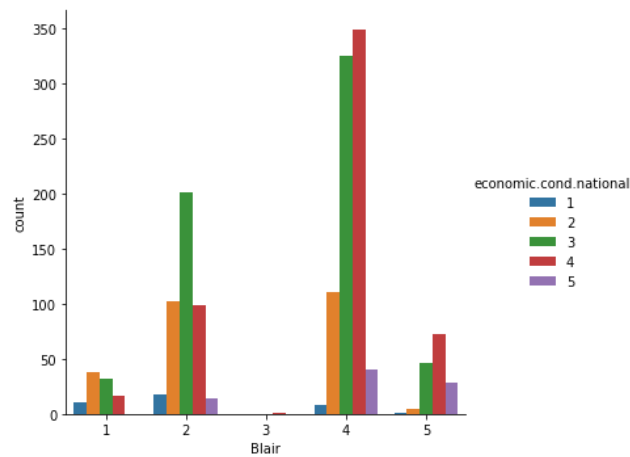
Inference

- There is very less correlation between the variables
- The highest positive correlation is seen between “economic_cond_national” and “economic_cond_household” (35%) with nearly similar results seen from “Blair” and “economic_cond_national” (35%)
- The highest negative correlation is seen between “Blair” and “Europe” (29%) with nearly similar results seen from “Blair” and “Hague” (24%)
- so, there is less or no chance of multi_collinearity **stripplot between "Hague" and age:**



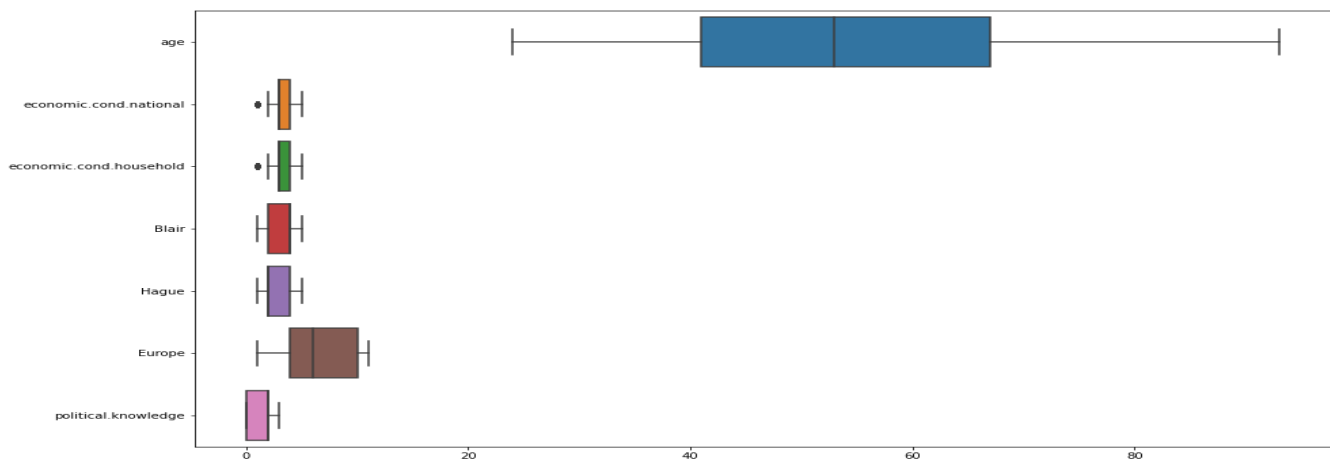
Above plot is a strip plot with jitter as True that really shows the distribution points on the assessment of the Conservative leader “Hague” on voters of various age. more voters are distributed in 2 and 4 group

Catplot Analysis - Blair(count) on economic.cond.national

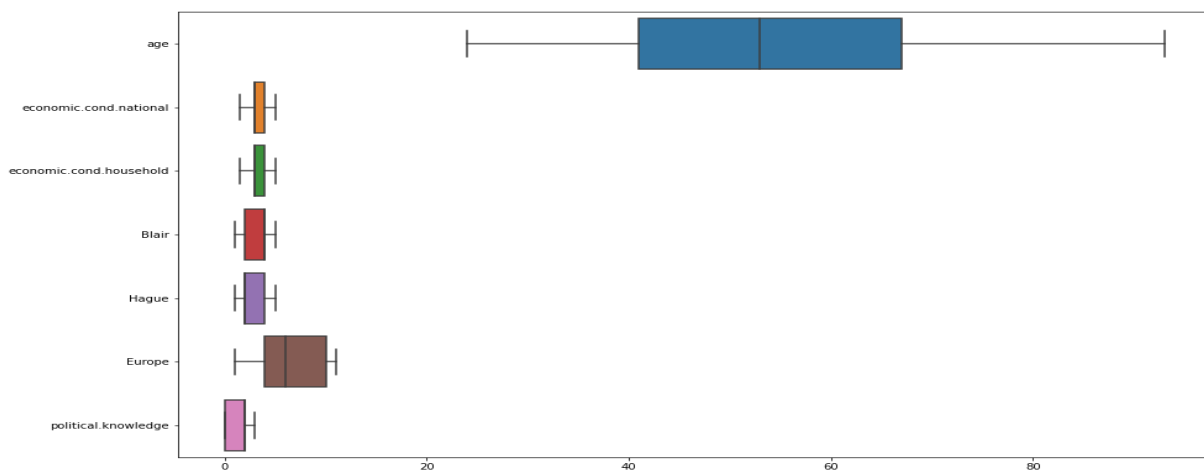


Assessment of current national economic conditions with Blair shows no 3 cluster have very less distribution whereas no 4 cluster have more distribution

Boxplot Before Outlier removal



After Outlier removal



All outliers are removed for further analysis

1.3 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).

Converting Object variables to categorical variables:

```
feature: vote
[Labour, Conservative]
Categories (2, object): [Conservative, Labour]
```

```
[1 0] feature:
gender
[female, male]
Categories (2, object): [female, male]
[0 1]
```

	vote	age	mic.cond.na	economic.cond.household	Blair	Hague	Europe	tical.knowled	gender
0	1	43	3	3	4	1	2	2	0
1	1	36	4	4	4	4	5	2	1
2	1	35	4	4	5	2	3	2	1
3	1	24	4	2	2	1	4	0	0
4	1	41	2	2	1	1	6	2	1

Inference

- Codes are an array of integers which are the positions of the actual values in the categories array.
- Here vote and gender are categorical variables are now converted into integers using codes
- All the variables in the data frame are integers

Scaling of the data

- Most of the times, your dataset will contain features highly varying in magnitudes, units, and range. But since, most of the machine learning algorithms use Euclidean distance between two data points in their computations, this is a problem.
- Differences in the scales across input variables may increase the difficulty of the problem being modelled.
- This means that you are transforming your data so that it fits within a specific scale, like 0-100 or 0-1
- Usually, the distance-based methods (E.g.: KNN) would require scaling as it is sensitive to extreme difference and can cause a bias.
- tree-based method uses split method (E.g.: Decision Trees) would not require scaling in general as it is unnecessary
- In this dataset, age is only continuous variable and rest of the variables have 1 to 5. Age variable is only scaled because it is continuous variable
- The method of scaling performed only on the 'age' variable is the Z-score scaling.
- Z-score scaling is the most common form of scaling that takes from the formula $(x - \text{mean}) / \text{standard deviation}$.

All the model prediction are done with scaled data

Train-Test Split

Separating independent (train) and dependent (test) variables for the linear regression model

X = independent (train) variables

Y = dependent (test) variables

The training set for the independent variables: (1061, 8)

The training set for the dependent variable: (1061, 1)

The test set for the independent variables: (456, 8)

The test set for the dependent variable: (456, 1)

Inference splitting the dataset into train and test set to build Logistic regression and LDA model (70:30) X_train :70% of data randomly chosen from the 8 columns. These are training independent variables

X_test :30% of data randomly chosen from the 8 columns. These are test independent variables

y_train :70% of data randomly chosen from the "vote" column. These are training dependent variables

y_test :30% of data randomly chosen from the "vote" columns. These are test dependent variables

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

Logistic Regression Model

Logistic regression is a fundamental classification technique. It belongs to the group of linear classifiers and is somewhat similar to polynomial and linear regression. It is the go-to method for binary classification problems (problems with two class values). Two libraries of Logistic regression

1. sklearn

2. statsmodel

Here for the model sklearn library is used

Applying GridSearchCV for Logistic Regression The probabilities on the training set

The probabilities on the test set

Column1	0	1
0	0.93016	0.06984
1	0.09863	0.90137
2	0.298616	0.701384
3	0.112255	0.887745
4	0.017899	0.982101

Column1	0	1
0	0.423273	0.576727
1	0.150187	0.849813
2	0.007369	0.992631
3	0.83337	0.16663
4	0.070077	0.929923

Fit the model to the training set

We now fit our model to the GridSearchCV for Logistic Regression model by training the model with our independent variable and dependent variables.

Inference

#Using GridsearchCV, we input various parameters like 'max_iter', 'penalty', 'solver', 'tol' which will help us to find best grid for prediction of the better model

#max_iter is an integer (100 by default) that defines the maximum number of iterations by the solver during model fitting.

#solver is a string ('liblinear' by default) that decides what solver to use for fitting the model. Other options are 'newton-cg', 'lbfgs', 'sag', and 'saga'.

#penalty is a string ('l2' by default) that decides whether there is regularization and which approach to use. Other options are 'l1', 'elasticnet', and 'none'.

#bestgrid: {'max_iter': 1000, 'penalty': 'l2', 'solver': 'saga', 'tol': 1e-05}

#Accuracy score of training data: 83.5%

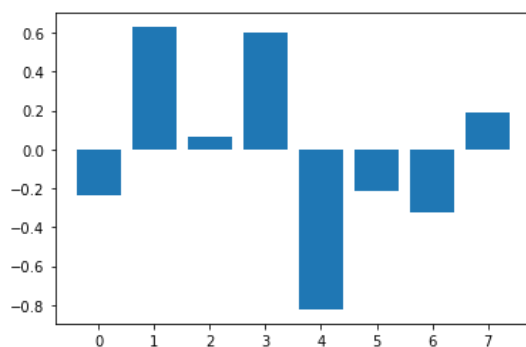
#Accuracy score of test data:83.5%

The coefficients for each of the independent attributes

The coefficient for age is -0.006854878831463495
The coefficient for economic.cond.national is 0.7135968821433287
The coefficient for economic.cond.household is 0.18721149727710462
The coefficient for Blair is 0.6703768181600023
The coefficient for Hague is -0.745697729263501
The coefficient for Europe is -0.183618187995406
The coefficient for political.knowledge is -0.25573979170938455
The coefficient for gender is 0.15672831586338198

Feature Importance Graphs

Feature: 0, Score: -0.23281
Feature: 1, Score: 0.62889
Feature: 2, Score: 0.06344
Feature: 3, Score: 0.60088
Feature: 4, Score: -0.82301
Feature: 5, Score: -0.21159
Feature: 6, Score: -0.32181 Feature:
7, Score: 0.19209



Inference

- The coefficients for each of the independent attributes
- The sign of a regression coefficient tells you whether there is a positive or negative correlation between each independent variable the dependent variable. A positive coefficient indicates that as the value of the independent variable increases, the mean of the dependent variable also tends to increase. A negative coefficient suggests that as the independent variable increases, the dependent variable tends to decrease.
- economic.cond.national have more positive coefficient . A positive coefficient indicates that as the value of the independent variable increases, the mean of the dependent variable also tends to increase the vote

Linear Discriminant Analysis

- Linear Discriminant Analysis (LDA) is a dimensionality reduction technique which is commonly used for the supervised classification problems.
- It is used for modeling differences in groups i.e., separating two or more classes. It is used to project the features in higher dimension space into a lower dimension space. ○ library used in LDA is sklearn

The probabilities on the training set

Column1	0	1
0	0.949216	0.050784
1	0.078241	0.921759
2	0.307389	0.692611
3	0.078963	0.921037
4	0.012161	0.987839

The probabilities on the test set

Column1	0	1
0	0.462093	0.537907
1	0.133955	0.866045
2	0.006414	0.993586
3	0.86121	0.13879
4	0.056545	0.943455

Applying GridSearchCV for LDA

- Using GridsearchCV, we input various parameters like 'max_iter', 'penalty', 'solver', 'tol' which will help us to find the best grid for prediction of the better model
- max_iter is an integer (100 by default) that defines the maximum number of iterations by the solver during model fitting.
- solver is a string ('liblinear' by default) that decides what solver to use for fitting the model. Other options are 'newton-cg', 'lbfgs', 'sag', and 'saga'.
- here 'solver': ['svd', 'lsqr', 'eigen'] are used with other parameters has default
- 'svd': Singular value decomposition (default). Does not compute the covariance matrix, therefore this solver is recommended for data with many features.
- 'lsqr': Least squares solution. Can be combined with shrinkage or custom covariance estimator.
- 'eigen': Eigenvalue decomposition. Can be combined with shrinkage or custom covariance estimator.
- bestgrid: {'solver': 'svd'}
- Training Data Class Prediction with a cut-off value of 0.5
- Test Data Class Prediction with a cut-off value of 0.5
- Accuracy score of training data: 83.4%
- Accuracy score of test data: 83.3%

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

KNN Model

KNN is a non-parametric and lazy learning algorithm. Non-parametric means there is no assumption for underlying data distribution. In KNN, K is the number of nearest neighbors. The number of neighbors is the core deciding factor

KNN has the following basic steps:

- Calculate distance
- Find closest neighbors
- Vote for labels we will be using popular scikit-learn package.
- The k-nearest neighbor algorithm is imported from the scikit-learn package.
- Create feature and target variables.
- Split data into training and test data.
- Generate a k-NN model using neighbors value.
- Train or fit the data into the model.

- Predict the future.

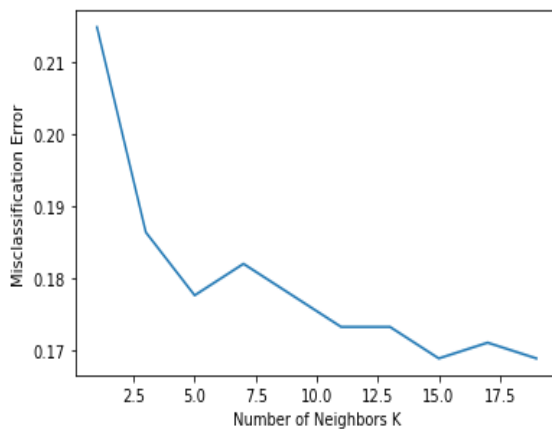
KNN classifier model.

- First, import the KNeighborsClassifier module and create KNN classifier object by passing argument number of neighbors in KNeighborsClassifier() function.
- Then, fit your model on the train set using fit() and perform prediction on the test set using predict().
- Let us build KNN classifier model for k=15.

misclassification error

```
[0.2149122807017544,
 0.1864035087719298,
 0.17763157894736847,
 0.18201754385964908,
 0.17763157894736847,
 0.17324561403508776,
 0.17324561403508776,
 0.16885964912280704,
 0.17105263157894735,
 0.16885964912280704]
```

Plot misclassification error vs k (with k value on X-axis)



The number of neighbors(K) in KNN is a hyperparameter that you need choose at the time of model building. You can think of K as a controlling variable for the prediction model.

n_neighbors = 15

The probabilities on the training set

Column1	0	1
0	0.933333	0.066667
1	0.133333	0.866667
2	0.266667	0.733333
3	0	1
4	0	1

Accuracy score of training data:84.7%

Accuracy score of test data:83.1%

The probabilities on the test set

Column1	0	1
0	0.666667	0.333333
1	0.266667	0.733333
2	0	1
3	0.733333	0.266667
4	0.133333	0.866667

Gaussian Naive Bayes

Naive Bayes is a classification technique based on the Bayes theorem. It is a simple but powerful algorithm for predictive modelling under supervised learning algorithms.

Gaussian Naive Bayes – This is a variant of Naive Bayes which supports continuous values and has an assumption that each class is normally distributed. All we would have to do is estimate the mean and standard deviation of the continuous variable. here we use library scikit-learn

Training Data Class Prediction with a cut-off value of 0.5

Test Data Class Prediction with a cut-off value of 0.5

The probabilities on the training set

Column1	0	1
0	0.984678	0.015322
1	0.065437	0.934563
2	0.271735	0.728265
3	0.080026	0.919974
4	0.007648	0.992352

Accuracy score of training data:83.5%

The probabilities on the test set

Column1	0	1
0	0.536792	0.463208
1	0.120285	0.879715
2	0.000332	0.999668
3	0.94524	0.05476
4	0.039267	0.960733

Accuracy score of test data:82.2%

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

Model tuning

Tuning is the process of maximizing a model's performance without overfitting or creating too high of a variance. In machine learning, this is accomplished by selecting appropriate "hyperparameters."

Hyperparameters can be thought of as the "dials" or "knobs" of a machine learning model. Choosing an appropriate set of hyperparameters is crucial for model accuracy, but can be computationally challenging. Hyperparameters differ from other model parameters in that they are not learned by the model automatically through training methods. Instead, these parameters must be set manually. Many methods exist for selecting appropriate hyperparameters.

Grid Search

Grid Search, also known as parameter sweeping, is one of the most basic and traditional methods of hyperparametric optimization. This method involves manually defining a subset of the hyperparametric space and exhausting all combinations of the specified hyperparameter subsets. Each combination's performance is then evaluated, typically using cross-validation, and the best performing hyperparametric combination is chosen. **Bagging with randomforest A Bagging classifier.**

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used to reduce the variance of a black-box estimator (e.g., RandomForest), by introducing randomization into its construction procedure and then making an ensemble out of it.

Bagging and random forests are “bagging” algorithms that aim to reduce the complexity of models that overfit the training data.

Bagging (Random Forest should be applied for Bagging) Inference
set the hyper parameters in randomforest classifier

N_estimators (only used in Random Forests) is the number of decision trees used in making the forest (default = 100).

Max_depth is an integer that sets the maximum depth of the tree. The default is None, which means the nodes are expanded until all the leaves are pure

Min_samples_split is the minimum number of samples required to split an internal node.

Min_samples_leaf defines the minimum number of samples needed at each leaf. The default input here is 1.

We now fit randomforest classifier model to the bagging model by training the model with our independent variable and dependent variables. At this point, you have the classification model defined.

The probabilities on the training set

Column1	0	1
0	0.765182	0.234818
1	0.181478	0.818522
2	0.372432	0.627568
3	0.117474	0.882526
4	0.060569	0.939431

Accuracy score of training data:83.6%

The probabilities on the test set

Column1	0	1
0	0.442012	0.557988
1	0.197157	0.802843
2	0.023259	0.976741
3	0.71091	0.28909
4	0.320904	0.679096

Accuracy score of test data:82.2%

Boosting

Boosting is an ensemble strategy that is consecutively builds on weak learners in order to generate one final strong learner. A weak learner is a model that may not be exactly accurate or may not take many predictors into account. By building a weak model, making conclusions about the various feature importance's and parameters, and then using those conclusions to build a new, stronger model, Boosting can effectively convert weak learners into a strong learner.

AdaBoost (Adaptive Boosting):

AdaBoost uses decision stumps as weak learners. A Decision Stump is a Decision Tree model that only splits off at one level, ergo the final prediction is based off only one feature. When AdaBoost makes its first Decision Stump, all observations are weighted evenly.

To correct previous error, when moving to the second Decision Stump, the observations that were classified incorrectly now carry more weight than the observations that were correctly classified. AdaBoost continues this strategy until the best classification model is built.

- GridSearchCV ADA boosting
- Using GridsearchCV, we input various parameters like {'algorithm', 'learning_rate', 'n_estimators'} which will helps us to find best grid for prediction of the better model
- N_estimators is the maximum number of estimators at which boosting is terminated. If a perfect fit is reached, the algo is stopped. The default here is 50.

- Learning_rate is the rate at which we are adjusting the weights of our model with respect to the loss gradient.
- The SAMME.R algorithm typically converges faster than SAMME, achieving a lower test error with fewer boosting iterations.
- bestgrid: {'algorithm': 'SAMME.R', 'learning_rate': 0.3, 'n_estimators': 51}

The probabilities on the training set

Column1	0	1
0	0.525024	0.474976
1	0.459948	0.540052
2	0.486375	0.513625
3	0.465187	0.534813
4	0.450924	0.549076

The probabilities on the test set

Column1	0	1
0	0.503244	0.496756
1	0.477533	0.522467
2	0.402849	0.597151
3	0.526377	0.473623
4	0.474901	0.525099

- Accuracy score of training data:84.1%
- Accuracy score of test data:82.0%

Gradient Boosting

Gradient Boosting for classification.

GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage $n_classes_$ regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function. Binary classification is a special case where only a single regression tree is induced.

- Using GridsearchCV, we input various parameters like {'criterion', 'loss', 'max_features', 'min_samples_split', 'n_estimators'} which will helps us to find best grid for prediction of the better model
- `loss{'deviance', 'exponential'}, default='deviance'`
The loss function to be optimized. 'deviance' refers to deviance (= logistic regression) for classification with probabilistic outputs.
- `criterion{'friedman_mse', 'mse', 'mae'}, default='friedman_mse'`
- The function to measure the quality of a split. Supported criteria are 'friedman_mse' for the mean squared error with improvement score by Friedman, 'mse' for mean squared error, and 'mae' for the mean absolute error.
- `n_estimatorsint, default=100`
- The number of boosting stages to perform. Gradient boosting is fairly robust to over-fitting so a large number usually results in better performance.
- `min_samples_splitint or float, default=2`
- The minimum number of samples required to split an internal node:If int, then consider min_samples_split as the minimum number.
- `max_features{'auto', 'sqrt', 'log2'}, int or float, default=None` ○ The number of features to consider when looking for the best split:
- If int, then consider max_features features at each split.

- best_params: {'criterion': 'friedman_mse', 'loss': 'exponential', 'max_features': 6, 'min_samples_split': 30, 'n_estimators': 51}
- best_estimator: GradientBoostingClassifier(loss='exponential', max_features=6, min_samples_split=30, n_estimators=51)

The probabilities on the training set

Column1	0	1
0	0.842479	0.157521
1	0.093991	0.906009
2	0.398639	0.601361
3	0.04832	0.95168
4	0.024671	0.975329

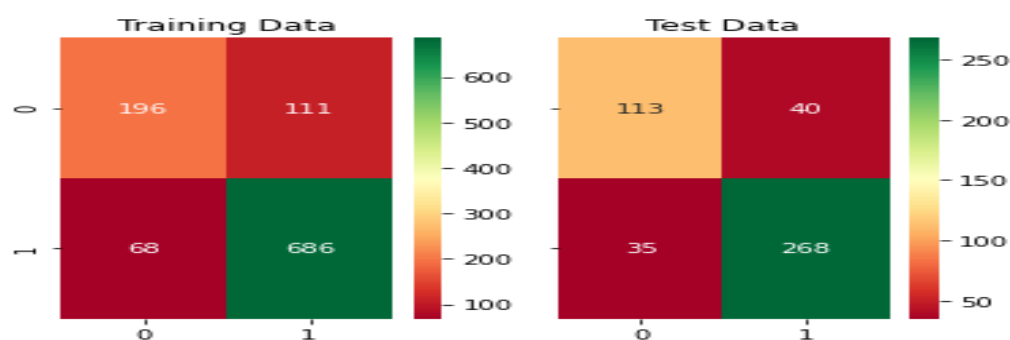
The probabilities on the test set

Column1	0	1
0	0.610818	0.389182
1	0.222798	0.777202
2	0.002342	0.997658
3	0.825742	0.174258
4	0.122592	0.877408

- Accuracy score of training data: 87.4%
- Accuracy score of test data: 83.3%

1.7 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model. Final Model: Compare the models and write inference which model is best/optimized.

Logistic Regression Model Confusion matrix on the training and test data



Inference

Training data:

True Negative : 196 False Positive : 111

False Negative : 68 True Positive : 686 Test data:

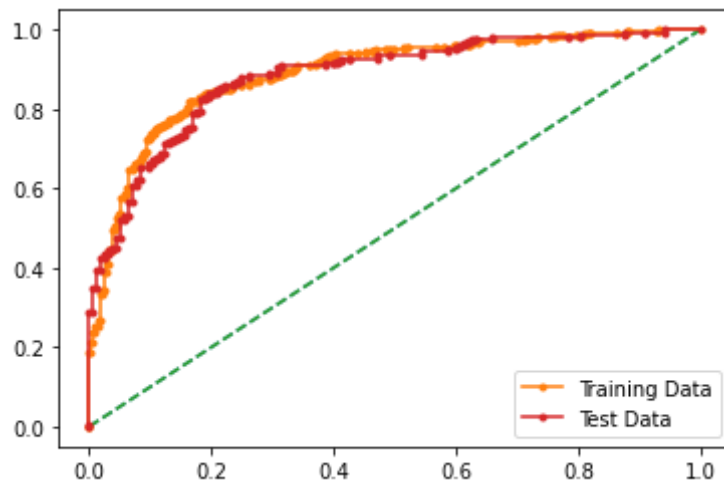
True Negative : 113 False Positive : 40 False

Negative : 35 True Positive : 268

Classification Report of training and test data

		precision	recall	f1-score	support
	0	0.74	0.64	0.69	307
1	0.86	0.91	0.88	0.89	754
	accuracy			0.83	
1061	macro avg	0.80	0.77	0.79	
1061	weighted avg	0.83	0.83	0.83	
1061					
		precision	recall	f1-score	support
	0	0.76	0.74	0.75	153
1	0.87	0.88	0.88	0.88	303
	accuracy			0.84	456
macro avg		0.82	0.81	0.81	456
weighted avg		0.83	0.84	0.83	456

AUC and ROC for the training and test data



AUC for the Training Data: 0.890

AUC for the Test Data: 0.883

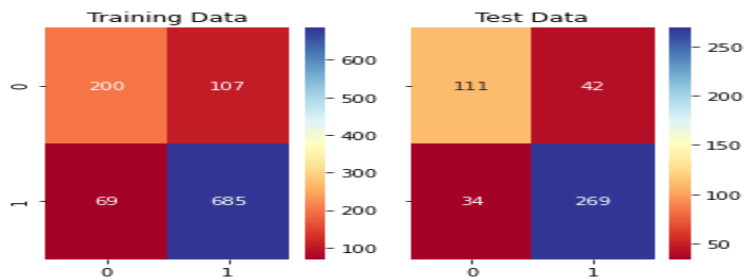
Inference

Train Data:

- AUC: 89%
 - Accuracy: 83%
 - precision : 86%
 - recall : 91%
 - f1 :88%
- Test Data:

- AUC: 88.3%
- Accuracy: 84%
- precision: 87%
- recall : 88%
- f1 : 88%
- Training and Test set results are almost similar, this proves no overfitting or underfitting

Linear Discriminant Analysis Confusion matrix on the training and test data



Inference:

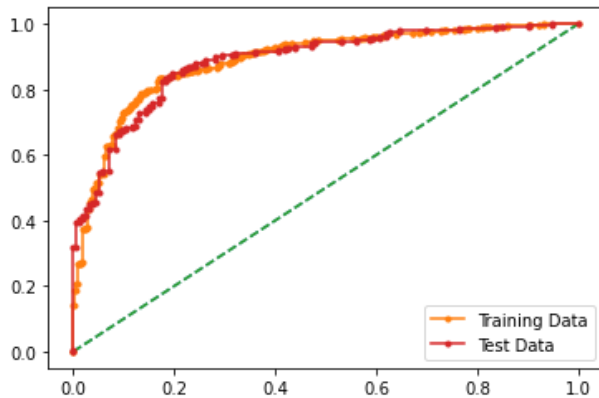
- Trainig data:
- True Negative : 200 False Positive : 107 ○ False Negative : 69 True Positive : 685 ○ Test data:
- True Negative : 111 False Positive : 42
- False Negative : 34 True Positive : 269

Classification Report of training and test data

	precision	recall	f1-score	support
0	0.74	0.65	0.69	307
1	0.86	0.91	0.89	754
accuracy			0.83	1061
macro avg	0.80	0.78	0.79	1061
weighted avg	0.83	0.83	0.83	1061

	precision	recall	f1-score	support
0	0.77	0.73	0.74	153
1	0.86	0.89	0.88	303
accuracy			0.83	456
macro avg	0.82	0.81	0.81	456
weighted avg	0.83	0.83	0.83	456

AUC and ROC for the training and test data



Inference

Train Data:

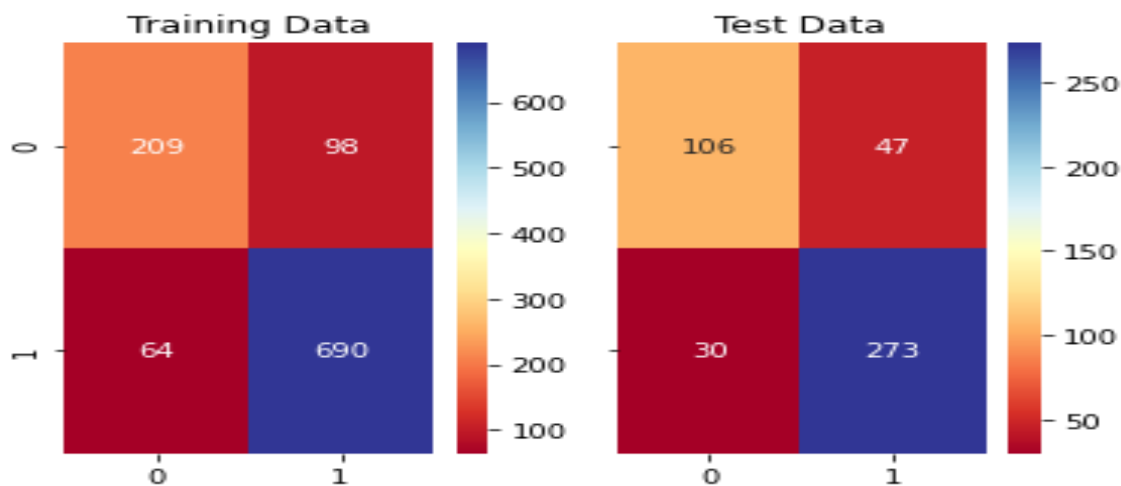
- AUC: 88.9%
- Accuracy: 83%
- precision : 86%
- recall : 91%
- f1 : 89%

Test Data:

- AUC: 88.8%
- Accuracy: 83%
- precision : 86%
- recall : 89%
- f1 : 88%

- Training and Test set results are almost similar, This proves no overfitting or underfitting

KNN Model Confusion matrix on the training and test data



Inference:

Training data:

- True Negative : 209 False Positive : 98
- False Negative : 64 True Positive : 690 Test data:
- True Negative : 116 False Positive : 47
- False Negative : 30 True Positive : 273

Classification Report of training and test data

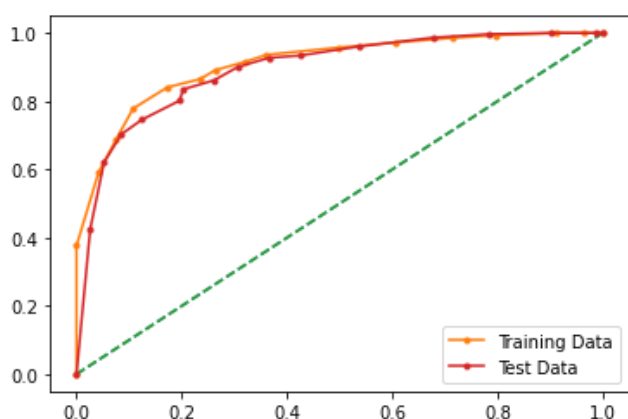
	precision	recall	f1-score	support
0	0.77	0.68	0.72	307
1	0.88	0.92	0.89	754

accuracy			0.85	1061
macro avg	0.82	0.80	0.81	1061
weighted avg	0.84	0.85	0.84	1061

	precision	recall	f1-score	support
0	0.78	0.69	0.73	153
1	0.85	0.90	0.88	303

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

AUC and ROC for the training and test data



AUC for the Training Data: 0.910

AUC for the Test Data: 0.893

Inference Train

Data:

- AUC: 91%
- Accuracy: 85%
- precision : 88%

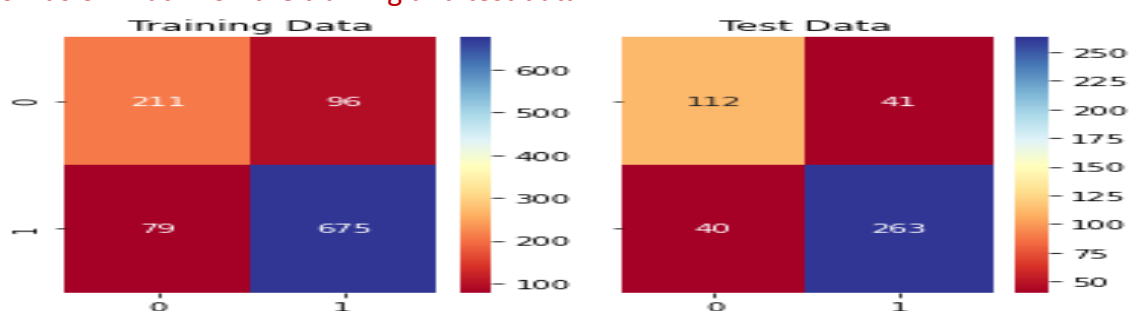
- recall : 92%
- f1 :89%

Test Data:

- AUC: 89.3%
- Accuracy: 83%
- precision :85%
- recall : 90%
- f1 : 88%
- Training and Test set results are almost similar,This proves no overfitting or underfitting

Naive Bayes model

Confusion matrix on the training and test data



Inference

Training data:

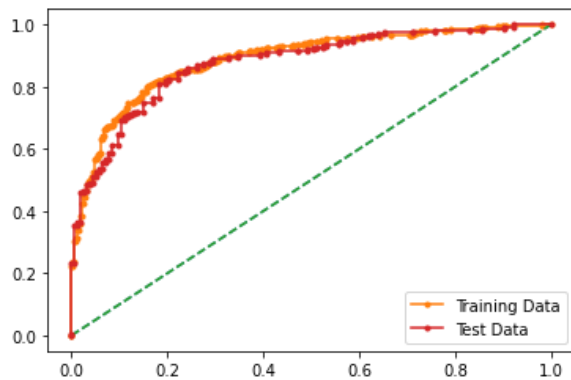
- True Negative : 211 False Positive : 96
- False Negative : 79 True Positive : 675
- True Negative : 112 False Positive : 41
- False Negative : 40 True Positive : 263

Classification Report of training and test data

	precision	recall	f1-score	support
0	0.73	0.69	0.71	307
1	0.88	0.90	0.89	754
accuracy			0.84	1061
macro avg	0.80	0.79	0.80	1061
weighted avg	0.83	0.84	0.83	1061

	precision	recall	f1-score	support
0	0.74	0.73	0.73	153
1	0.87	0.87	0.87	303
accuracy			0.82	
456 macro avg	0.80	0.80	0.80	
456 weighted avg	0.82	0.82	0.82	
456				

AUC and ROC for the training and test data



AUC for the Training Data: 0.888

AUC for the Test Data: 0.876

Inference

Train Data:

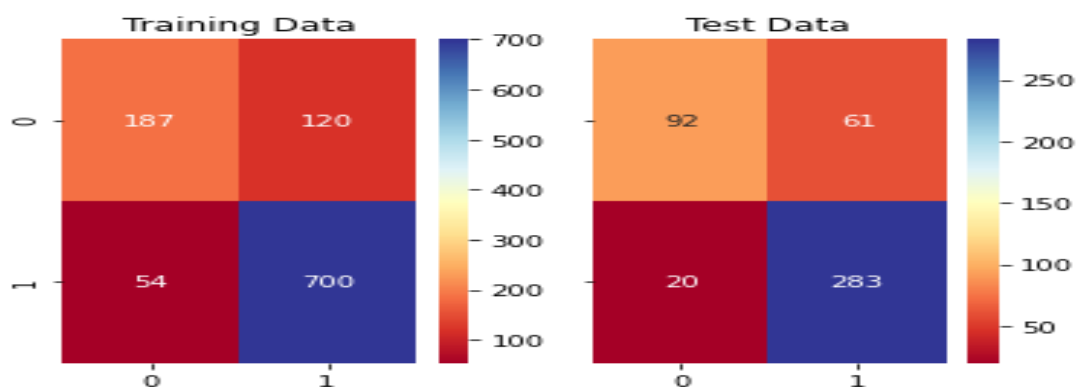
- AUC: 88.8%
- Accuracy: 84%
- precision : 88%
- recall : 90%
- f1 :89%

Test Data:

- AUC: 87.6%
- Accuracy: 82%
- precision :88%
- recall : 88%
- f1 : 88%
- Training and Test set results are almost similar,This proves no overfitting or underfitting

Bagging with random forest

Confusion matrix on the training and test data



Inference Training

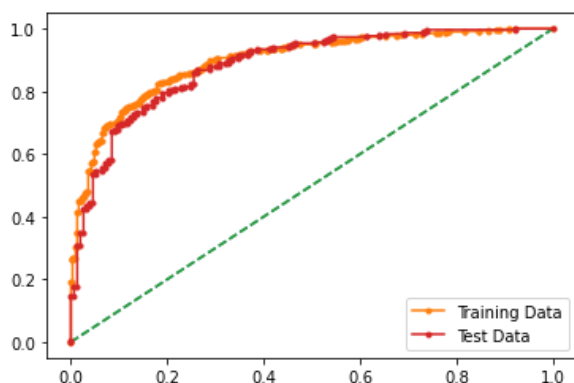
data:

- True Negative : 187 False Positive : 120
- False Negative : 54 True Positive : 700 Test data:
- True Negative : 92 False Positive : 61
- False Negative : 20 True Positive : 283

Classification Report of training and test data

		precision	recall	f1-score	support
	0	0.78	0.61	0.68	307
	1	0.85	0.93	0.89	754
	accuracy			0.84	
1061	macro avg	0.81	0.77	0.79	
1061	weighted avg	0.83	0.84	0.83	
1061					
		precision	recall	f1-score	support
	0	0.82	0.60	0.69	153
	1	0.82	0.93	0.87	303
	accuracy			0.82	456
	macro avg	0.82	0.77	0.78	456
	weighted avg	0.82	0.82	0.81	456

AUC and ROC for the training and test data



AUC for the Training Data: 0.897

AUC for the Test Data: 0.884

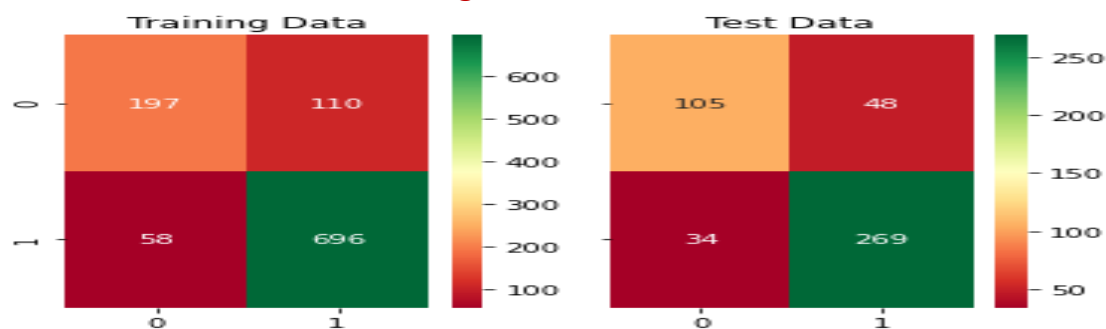
Inference Train

Data:

- AUC: 89.7%
- Accuracy: 84%
- precision : 85%
- recall : 93%
- f1 :89% Test Data:
- AUC: 88.4%
- Accuracy: 82%
- precision :82%
- recall : 91%
- f1 : 87%
- Training and Test set results are almost similar,This proves no overfitting or underfitting

AdaBoostClassifier

Confusion matrix on the training and test data



Inference Training data:

- True Negative : 197 False Positive : 110
- False Negative : 58 True Positive : 696 Test data:
- True Negative : 105 False Positive : 48
- False Negative : 34 True Positive : 269

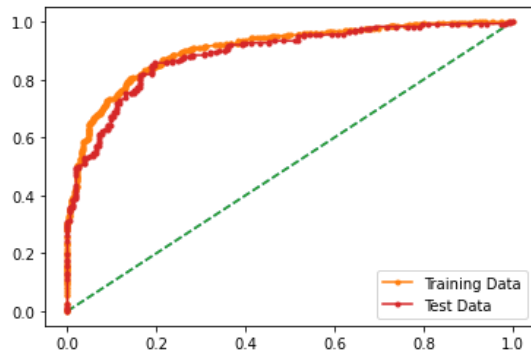
Classification Report of training and test data

		precision	recall	f1-score	support
	0	0.77	0.64	0.70	307
	1	0.86	0.92	0.89	754
accuracy				0.84	1061
macro avg		0.82	0.78	0.80	1061
weighted avg		0.84	0.84	0.84	1061

		precision	recall	f1-score	support
0	0.76	0.69	0.72	153	
1	0.85	0.89	0.87	303	

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

AUC and ROC for the training and test data



AUC for the Training Data: 0.906

AUC for the Test Data: 0.889

Inference

Train Data:

- AUC: 90.6%
- Accuracy: 84%
- precision : 85%
- recall : 92%
- f1 :89%

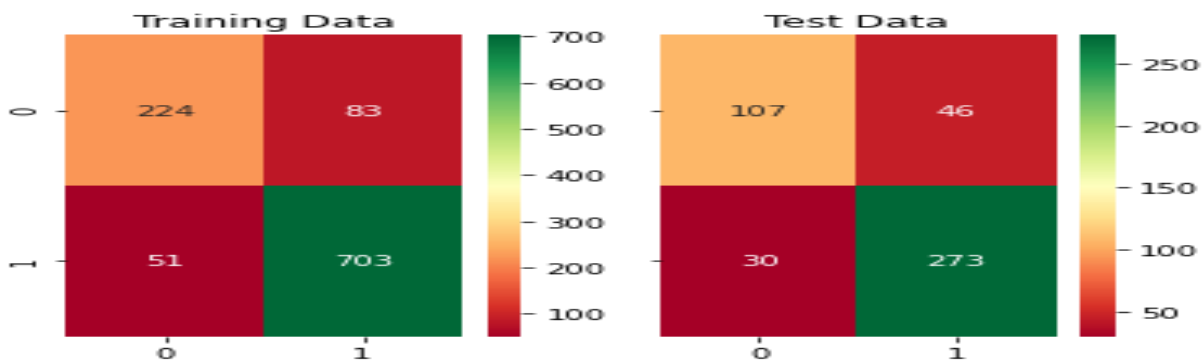
Test Data:

- AUC: 88.9%
- Accuracy: 82%
- precision :85%
- recall : 89%
- f1 : 87%

- Training and Test set results are almost similar,This proves no overfitting or underfitting

Gradient Boosting

Confusion matrix on the training and test data



Inference

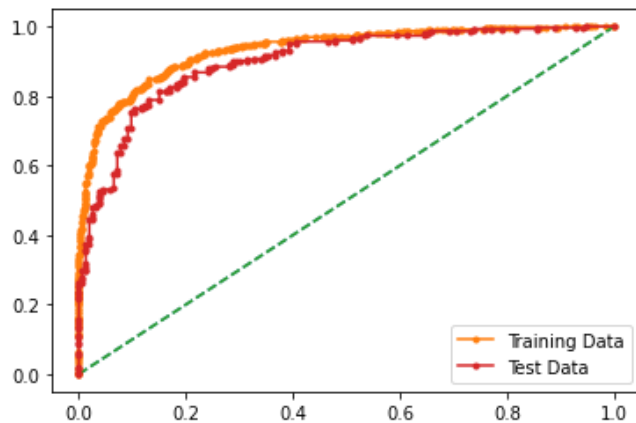
Training data:

- True Negative : 224 False Positive : 83
- False Negative : 51 True Positive : 703
- True Negative : 107 False Positive : 46
- False Negative : 30 True Positive : 273

Classification Report of training and test data

		precision	recall	f1-score	support
	0	0.81	0.73	0.77	307
	1	0.89	0.93	0.91	754
	accuracy			0.87	
1061	macro avg	0.85	0.83	0.84	
1061	weighted avg	0.87	0.87	0.87	
1061					
		precision	recall	f1-score	support
	0	0.78	0.70	0.74	153
	1	0.86	0.90	0.88	303
	accuracy			0.83	456
	macro avg	0.82	0.80	0.81	456
	weighted avg	0.83	0.83	0.83	456

AUC and ROC for the training and test data



AUC for the Training Data: 0.934

AUC for the Test Data: 0.901 **Inference**

Train Data:

- AUC: 93.4%
- Accuracy: 87%
- precision : 89%
- recall : 93%
- f1 :91% **Test Data:**

- AUC: 90.1%
- Accuracy: 83%
- precision :86%
- recall : 90%
- f1 : 88%
- Training and Test set results are almost similar,This proves no overfitting or underfitting
- This Gradient boosting shown better result for this dataset

Final Model: Comparing all the models

	LR Train	LR Test	LDA Train	LD A Test	KN N Train	KN N Test	NB Train	NB Test	BAGGI NG Train	BAGGI NG Test	AD A Train	AD A Test	Gradient Train	Gradient Test
Accuracy	0.83	0.84	0.83	0.83	0.85	0.83	0.84	0.82	0.84	0.82	0.84	0.82	0.87	0.87
AUC	0.89	0.88	0.9	0.88	0.91	0.89	0.89	0.88	0.9	0.88	0.91	0.89	0.93	0.9
Recall	0.91	0.88	0.91	0.89	0.92	0.9	0.9	0.87	0.93	0.93	0.92	0.89	0.93	0.9
Precision	0.86	0.87	0.86	0.86	0.88	0.85	0.88	0.87	0.85	0.82	0.86	0.85	0.89	0.86
F1 Score	0.88	0.88	0.89	0.88	0.89	0.88	0.89	0.87	0.89	0.87	0.89	0.87	0.91	0.88

Inference

- Almost all the models performed well with accuracy between 82% to 84%. Gradient boosting improved the accuracy to 87% so it is better model for to predict which party a voter will vote
- Comparing all the model ,Gradient boosting model is best model for this dataset with accuracy of 87% in both training and test set
- AUC of Train and test in Gradient boosting model is 93% and 90% respectively
- f1 score of Train and test in Gradient boosting model is 91% and 88% respectively
- Precision of Train and test in Gradient boosting model is 89% and 86% respectively
- Recall of Train and test in Gradient boosting model is 93% and 90% respectively
- Accuracy ,AUC,Precision,Recall for test data are almost in line with training data in Gradient boosting model.This indicates no overfitting or underfitting in the model
- Gradient boosting improved the accuracy to 87% so it is better model for to predict which party a voter will vote

Overall Best/Optimized model:

Almost all the models performed well with accuracy between 82% to 84% with scaled data. But Gradient boosting is best and optimised model with accuracy of 87% and also best AUC,Precision,f1 score, Recall .

1.8 Based on these predictions, what are the insights?

The main business objective of this project is to build a model to predict which party a voter will vote for based on the given information, to create an exit poll that will help in predicting overall win and seats covered by a particular party.

Various model was built on scaled dataset in that it is found that Gradient Boosting model gave best/optimized accuracy with 87% to predict which party a voter will vote based on given information and clearly an exit poll can be built that can help in overall win and seats covered by a particular party.

Sample data test

With sample data we can test against each optimised model.

Information of the sample data:

- age':4
- 'economic.cond.national': 6
- 'economic.cond.household': 10
- 'Blair': 8
- 'Hague':20
- 'Europe': 11
- 'political_knowledge': 7
- 'gender': 21

From the sample voters , final model conclusion

Model	Prediction
Logistic Regression	conservative party
Linear Discriminant Analysis	conservative party
K-Nearest Neighbour	conservative party
Naive Bayes	labour party
Bagging(with Random Forest)	labour party
Adaptive Boosting	conservative party
Gradient Boosting	conservative party

Conclusion

Based on this machine learning models, we can predict which party the voter might vote with more sample voters. Exit polls can be created with this model to predict which party will win or lose and seats covered by a particular party.

Problem2

**Text analysis on 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

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

(Hint: use `.words()`, `.raw()`, `.sent()` for extracting counts)

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

characters in Roosevelt speech

The number of characters in Roosevelt speech are: 7571

**characters in Kennedy
speech**

The number of characters in Kennedy speech are: 7618

**characters in Nixon
speech**

The number of characters in Nixon speech are: 9991

Words in Roosevelt speech

The number of Words in Roosevelt speech are: 1536

Words in Kennedy speech

The number of Words in Kennedy speech are: 1546

Words in Nixon speech

The number of Words in Nixon speech are: 2028

**sentences in Roosevelt
speech**

The number of sentences in Roosevelt speech are: 68

**sentences in Kennedy
speech**

The number of sentences in Kennedy speech are: 52

sentences in Nixon speech

The number of sentences in Nixon speech are: 69

2.2 Remove all the stopwords from all three speeches

In natural language processing, useless words (data), are referred to as stop words.

Stop Words:

A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. We would not want these words to take up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to stop words.

Libraries used: from nltk.corpus

```
import stopwords
```

```
nltk.download('stopwords')
```

Stopwords removal is done through “from nltk.corpus import stopwords”.

Stemming is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the Language. **Library used:**

```
from nltk.stem.porter import PorterStemmer
```

Most common words in Roosevelt speech after removing stopwords

Most common words in Roosevelt speech after removing stopwords

```
['nation', 'know', 'peopl', 'spirit', 'life', 'democraci', 'us', 'america', 'live', 'year', 'human', 'freedom', 'measur', 'men', 'govern', 'new', 'bodi', 'mind', 'speak', 'day', 'state', 'american', 'must', 'someth', 'faith', 'unit', 'task', 'preserv', 'within', 'histori', 'three', 'form', 'futur', 'seem', 'hope', 'understand', 'thing', 'free', 'alon', 'still', 'everi', 'contin', 'like', 'person', 'world', 'sacr', 'word', 'came', 'land', 'first']
```

Most common words in Kennedy speech after removing stopwords

Most common words in Kennedy speech after removing stopwords

```
['let', 'us', 'power', 'world', 'nation', 'side', 'new', 'pledg', 'ask', 'citizen', 'peac', 'shall', 'free', 'final', 'presid', 'fellow', 'freedom', 'begin', 'man', 'hand', 'human', 'first', 'gener', 'american', 'war', 'alway', 'know', 'support', 'unit', 'cannot', 'hope', 'help', 'weak', 'arm', 'countri', 'call', 'today', 'well', 'god', 'form', 'poverti', 'life', 'globe', 'right', 'state', 'dare', 'word', 'go', 'friend', 'bear']
```

Most common words in Nixon speech after removing stopwords

Most common words in Nixon speech after removing stopwords

```
['us', 'let', 'america', 'peac', 'world', 'respons', 'new', 'nation', 'govern', 'gr  
eat', 'year', 'home', 'abroad', 'make', 'togeth', 'shall', 'time', 'polici', 'role'  
, 'right', 'everi', 'histori', 'better', 'come', 'respect', 'peopl', 'live', 'help'  
, 'four', 'war', 'today', 'era', 'progress', 'other', 'build', 'act', 'challeng', '  
one', 'mr', 'share', 'meet', 'promis', 'long', 'work', 'preserv', 'freedom', 'place'  
, 'system', 'god', 'way']
```

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

Top three words in Roosevelt's speech(after removing the stopwords)

```
[('nation', 17), ('know', 10), ('peopl', 9)]
```

Top three words in Kennedy's speech(after removing the stopwords)

```
[('let', 16), ('us', 12), ('power', 9)]
```

Top three words in Nixon's speech(after removing the stopwords)

```
[('us', 26), ('let', 22), ('america', 21)]
```

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

Wordcloud

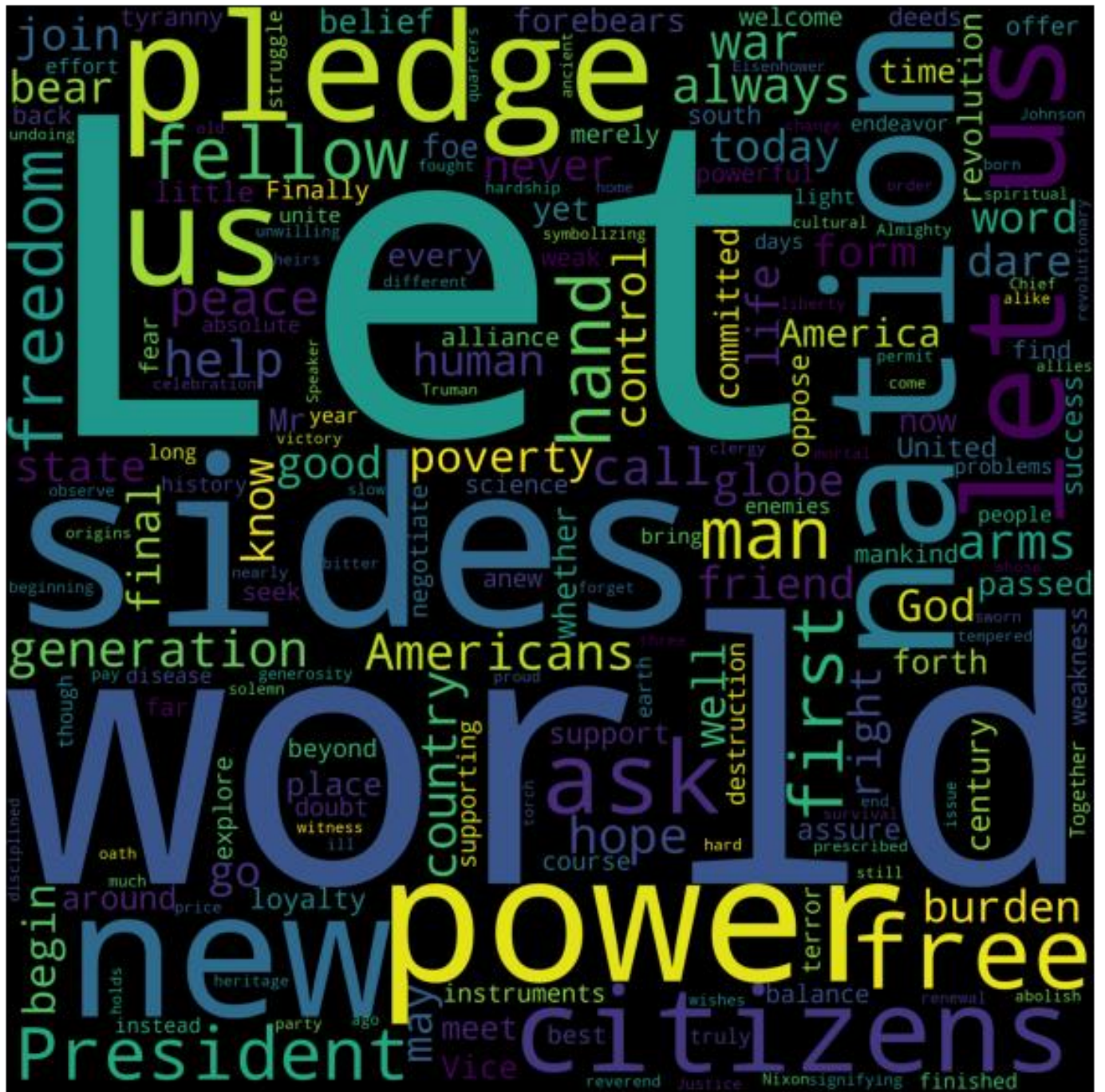
Many times you might have seen a cloud filled with lots of words in different sizes, which represent the frequency or the importance of each word. This is called [Tag Cloud](#) or WordCloud. **Liabrary used**

```
from wordcloud import WordCloud
```

Roosevelt speech

```
Word Cloud (after cleaning)!!
```


Kennedy speech



Inference:

- Most Frequent words are let, world, slides, power
- Less Frequent words are best, wishes, slow

Nixon speech



Inference:

- Most Frequent words are America, let, us, nation
- Less Frequent words are flimsy, adopted, saw

Conclusion:

This project data presented from '1941-Roosevelt.txt', '1961-Kennedy.txt' and '1973-Nixon.txt', we analysed some interesting insights like the number of characters, words, and sentences from the speeches. To Identify the strength and the sentiment of these presidential speeches the stop words were removed (punctuation and lowering the characters were removed) along with stemming. We analysed some of the common words from their speeches which inspired many Americans. word cloud method which visually show most common words to least common words