BUSINESS REPORT

PROBLEM – 1

Objective:- In this project we are working on elections database and need to work on building classification model and predicting which party a voter will vote based on various parameters of an individual.

PART-1 -- Data Ingestion:

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

Solution: - To start with the analysis lets load the data and look at the sample data, and perform basic check. Top 5 rows of 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

Descriptive Analysis of Dataset: -

- Getting info of dataset: -

```
 #   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
```

- Getting shape of dataset: -

```
The shape of dataset is :(1525, 9)
```

- Description of dataset with statistical inferences: -

	count	mean	std	min	25%	50%	75%	max
age	1525.0	54.182295	15.711209	24.0	41.0	53.0	67.0	93.0
economic.cond.national	1525.0	3.245902	0.880969	1.0	3.0	3.0	4.0	5.0
economic.cond.household	1525.0	3.140328	0.929951	1.0	3.0	3.0	4.0	5.0
Blair	1525.0	3.334426	1.174824	1.0	2.0	4.0	4.0	5.0
Hague	1525.0	2.746885	1.230703	1.0	2.0	2.0	4.0	5.0
Europe	1525.0	6.728525	3.297538	1.0	4.0	6.0	10.0	11.0
political.knowledge	1525.0	1.542295	1.083315	0.0	0.0	2.0	2.0	3.0

- Description of dataset including variables with object datatypes: -

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
vote	1525	2	Labour	1063	NaN	NaN	NaN	NaN	NaN	NaN	NaN
age	1525.0	NaN	NaN	NaN	54.182295	15.711209	24.0	41.0	53.0	67.0	93.0
economic.cond.national	1525.0	NaN	NaN	NaN	3.245902	0.880969	1.0	3.0	3.0	4.0	5.0
economic.cond.household	1525.0	NaN	NaN	NaN	3.140328	0.929951	1.0	3.0	3.0	4.0	5.0
Blair	1525.0	NaN	NaN	NaN	3.334426	1.174824	1.0	2.0	4.0	4.0	5.0
Hague	1525.0	NaN	NaN	NaN	2.746885	1.230703	1.0	2.0	2.0	4.0	5.0
Europe	1525.0	NaN	NaN	NaN	6.728525	3.297538	1.0	4.0	6.0	10.0	11.0
political.knowledge	1525.0	NaN	NaN	NaN	1.542295	1.083315	0.0	0.0	2.0	2.0	3.0
gender	1525	2	female	812	NaN	NaN	NaN	NaN	NaN	NaN	NaN

- Checking for Null values: -

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

- Checking number of duplicate rows: -

```
Number of duplicate rows: 8
```

- Deleted duplicate rows as a part of data pre-processing: -

```
Number of duplicate rows after deleting duplicates: 0
```

- Checking shape of data after removing duplicate rows: -

```
The shape of dataset is :(1517, 9)
```

- Checking standard Deviation of data: -

```
age                    15.701741
economic.cond.national  0.881792
economic.cond.household 0.931069
Blair                   1.174772
Hague                   1.232479
Europe                  3.299043
political.knowledge     1.084417
```

- Checking Kurtosis and Skewness of data: -

	Kurtosis	Skewness
age	-0.94	0.14
economic.cond.national	-0.26	-0.24
economic.cond.household	-0.21	-0.14
Blair	-1.06	-0.54
Hague	-1.40	0.15
Europe	-1.24	-0.14
political.knowledge	-1.22	-0.42

INFERENCES FROM ABOVE DESCRIPTIVE STATISTICS ANALYSIS

- ✓ The dataset contains 10 variables, 9 independent and 1 dependent variable and 1525 observations (rows). Below are the details of variables: -
  1. vote: Party choice: Conservative or Labour – This variable is of object data type
  2. age: in years – This variable is of int data type.
  3. economic.cond.national: Assessment of current national economic conditions, 1 to 5. – This variable is of int data type.
  4. economic.cond.household: Assessment of current household economic conditions, 1 to 5. – This variable is of int data type.
  5. Blair: Assessment of the Labour leader, 1 to 5. – This variable is of int data type.
  6. Hague: Assessment of the Conservative leader, 1 to 5. – This variable is of int data type.
  7. Europe: an 11-point scale that measures respondents' attitudes toward European integration. High scores represent 'Eurosceptic' sentiment. – This variable is of int data type.
  8. political.knowledge: Knowledge of parties' positions on European integration, 0 to 3. – This variable is of int data type.
  9. gender: female or male. – This variable is of int data type.
- ✓ Variable "age" is continuous in nature and all other 9 variables are categorical in nature.
- ✓ First column has Sl. No, Hence, deleted as it won't be significant in further data analysis. After deleted first column the final data set has 9 variables. 8 independent and 1 dependent.
- ✓ The variables "vote" and "gender" are of object data types remaining all other variables are of numeric/int data types.
- ✓ Data set doesn't have any null values, we have observations with "0" however those values are logical and shouldn't be altered.
- ✓ There are 8 duplicated observations. Deleting them resulted with remaining observations/rows of 1517.
- ✓ None of the variable shows normal distribution. The variables "age" and "Hague" are right skewed and rest of the variables are left skewed.
- ✓ Every variable has different standard deviation.
- ✓ All the variable shows Platykurtic form for Kurtosis.

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

Solutions: -

Univariate Analysis: -

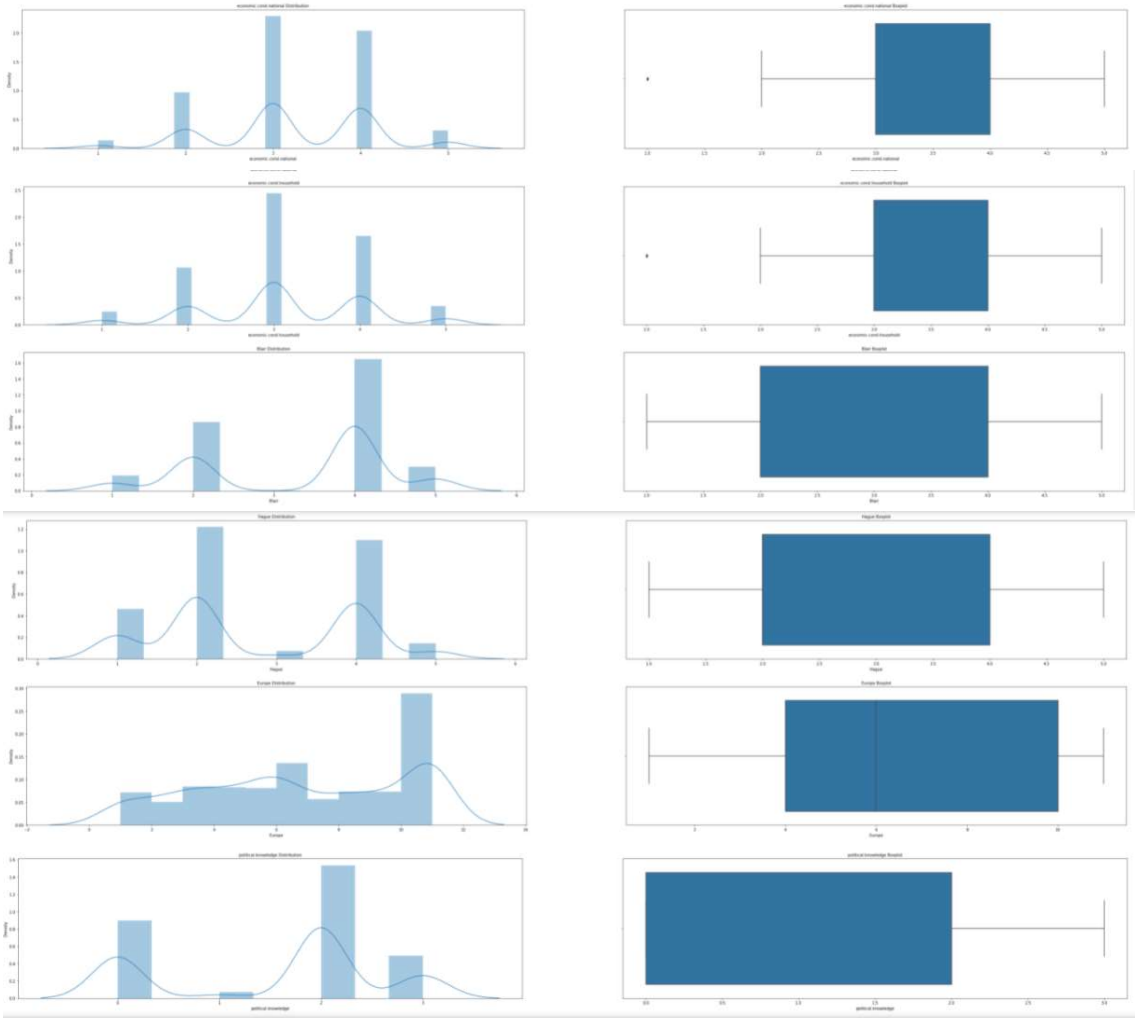
- ➢ Checking percentage of outliers for each variable in data: -

	outlier %
Blair	0.00
Europe	0.00
Hague	0.00
age	0.00
economic.cond.household	4.28
economic.cond.national	2.44
gender	0.00
political.knowledge	0.00
vote	0.00

- Summary stats of dataset: -

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
vote	1525	2	Labour	1063	NaN	NaN	NaN	NaN	NaN	NaN	NaN
age	1525.0	NaN	NaN	NaN	54.182295	15.711209	24.0	41.0	53.0	67.0	93.0
economic.cond.national	1525.0	NaN	NaN	NaN	3.245902	0.880969	1.0	3.0	3.0	4.0	5.0
economic.cond.household	1525.0	NaN	NaN	NaN	3.140328	0.929951	1.0	3.0	3.0	4.0	5.0
Blair	1525.0	NaN	NaN	NaN	3.334426	1.174824	1.0	2.0	4.0	4.0	5.0
Hague	1525.0	NaN	NaN	NaN	2.746885	1.230703	1.0	2.0	2.0	4.0	5.0
Europe	1525.0	NaN	NaN	NaN	6.728525	3.297538	1.0	4.0	6.0	10.0	11.0
political.knowledge	1525.0	NaN	NaN	NaN	1.542295	1.083315	0.0	0.0	2.0	2.0	3.0
gender	1525	2	female	812	NaN	NaN	NaN	NaN	NaN	NaN	NaN

- Box Plots and Distribution Plots: -



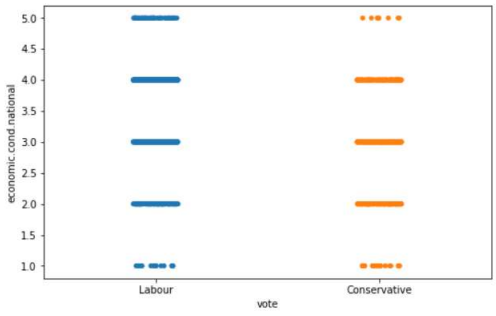
- Null values in dataset: -

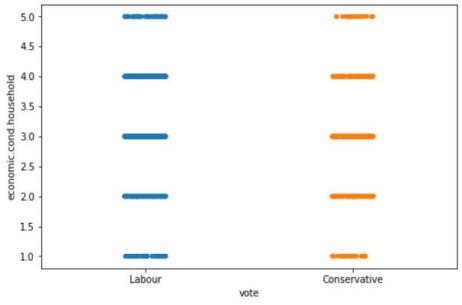
```
vote                        0
age                         0
economic.cond.national      0
economic.cond.household     0
Blair                       0
Hague                       0
Europe                      0
political.knowledge         0
gender                      0
```

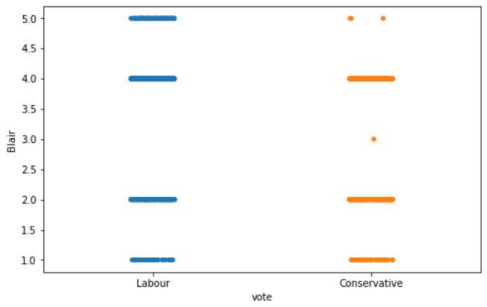
- Shape of dataset: -

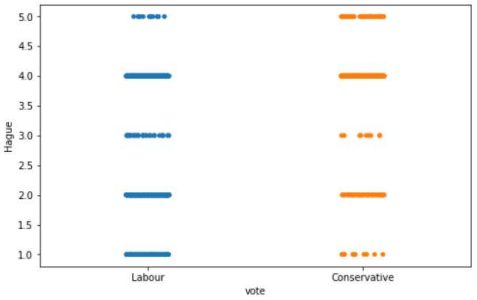
```
The shape of dataset is :(1525, 9)
```

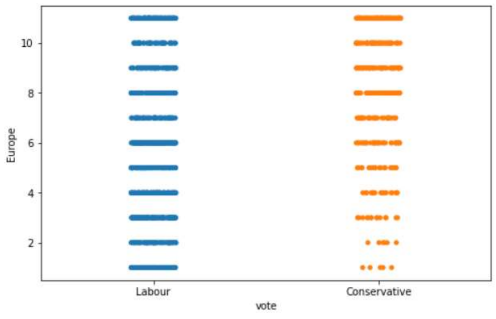
Bivariate Analysis: -

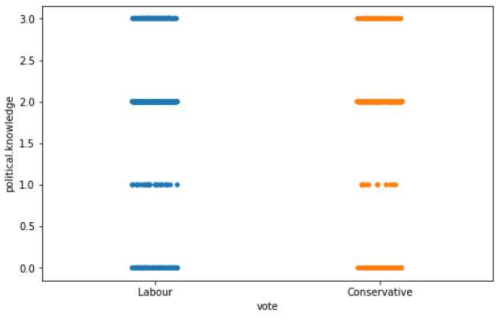


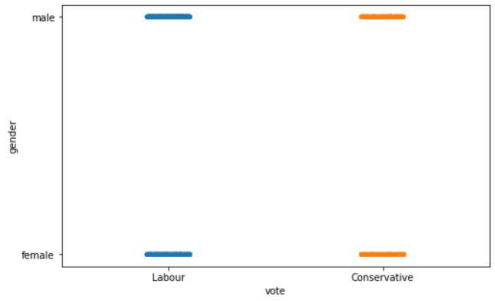






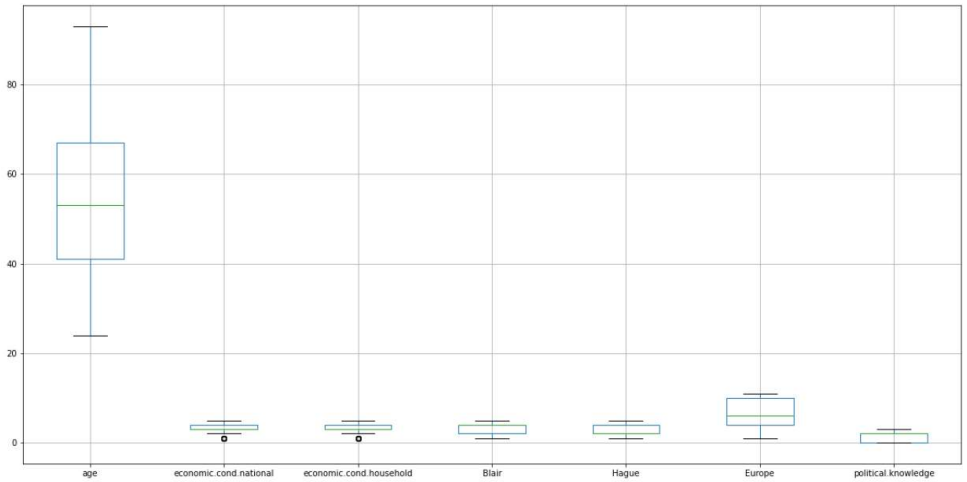




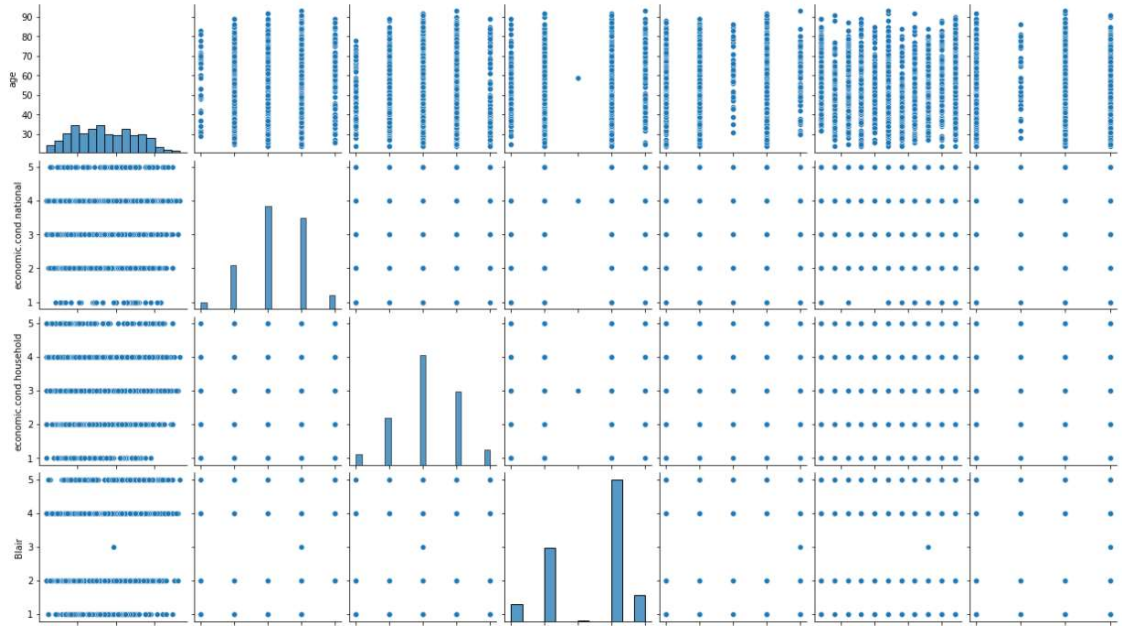


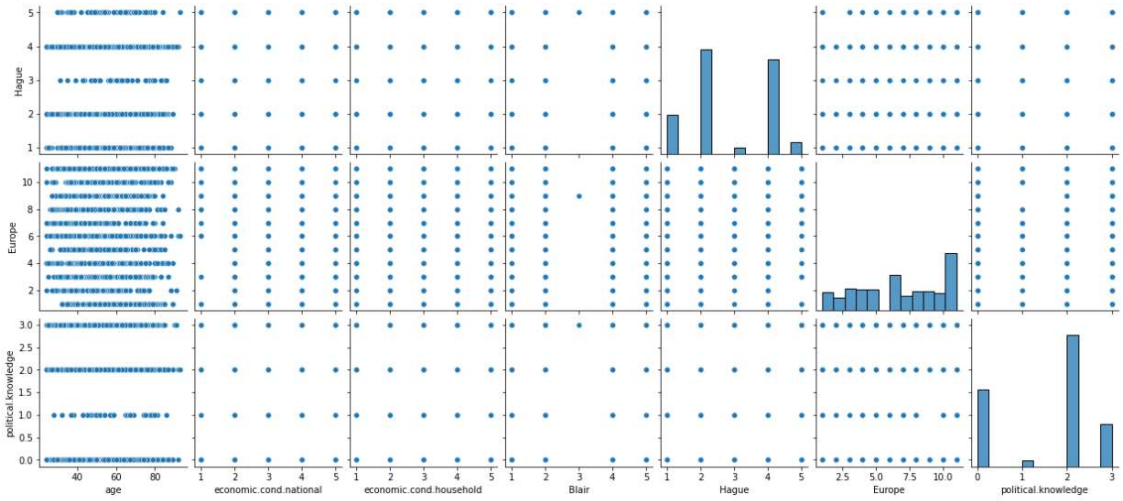
Multi-Variate Analysis: -

- Boxplot for checking outliers for all variables: -



- Pair-Plots for multiple variables





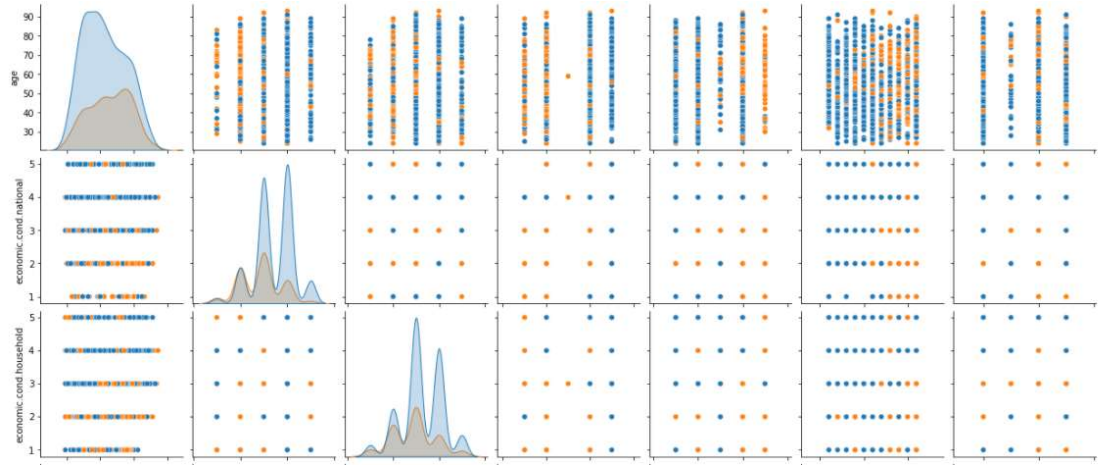
- Checking co-variance of multiple variables: -

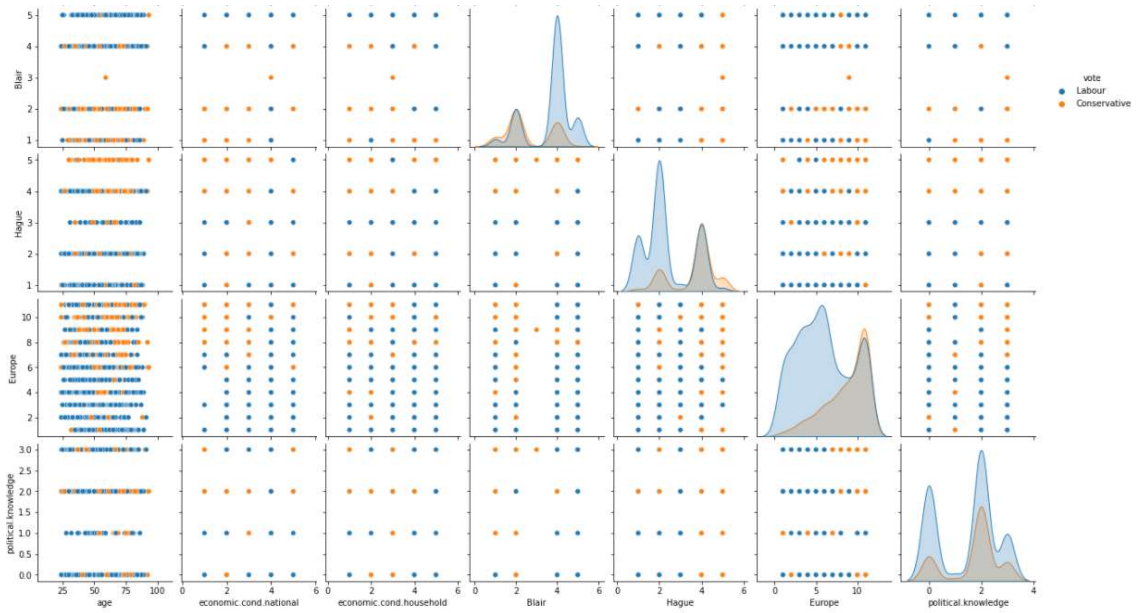
	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge
age	246.544655	0.258740	-0.568222	0.591818	0.602692	3.344366	-0.793429
economic.cond.national	0.258740	0.777558	0.285454	0.337851	-0.218216	-0.608432	-0.022481
economic.cond.household	-0.568222	0.285454	0.866890	0.236065	-0.115202	-0.346780	-0.038900
Blair	0.591818	0.337851	0.236065	1.380089	-0.352571	-1.146966	-0.027134
Hague	0.602692	-0.218216	-0.115202	-0.352571	1.519005	1.161811	-0.039970
Europe	3.344366	-0.608432	-0.346780	-1.146966	1.161811	10.883687	-0.540915
political.knowledge	-0.793429	-0.022481	-0.038900	-0.027134	-0.039970	-0.540915	1.175961

- Checking correlation of multiple variables: -

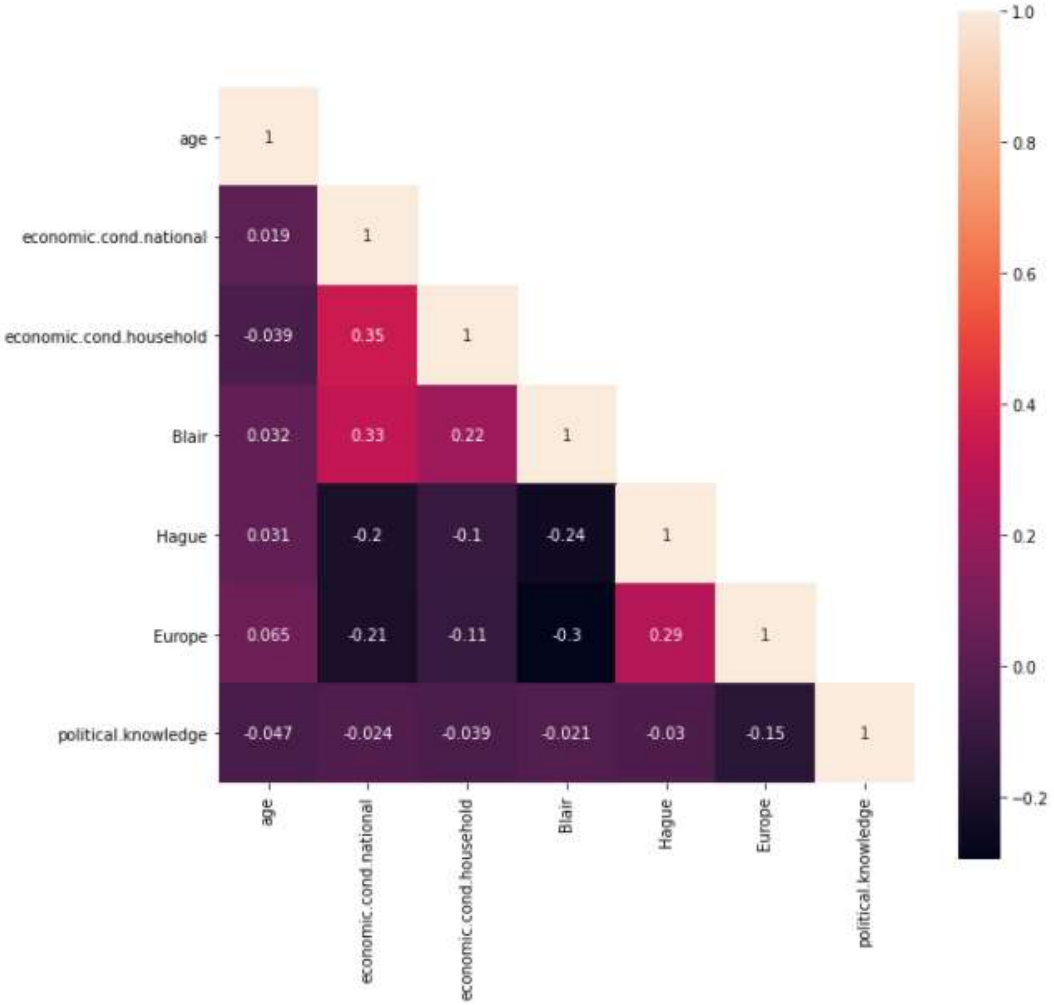
	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge
age	1.000000	0.018687	-0.038868	0.032084	0.031144	0.064562	-0.046598
economic.cond.national	0.018687	1.000000	0.347687	0.326141	-0.200790	-0.209150	-0.023510
economic.cond.household	-0.038868	0.347687	1.000000	0.215822	-0.100392	-0.112897	-0.038528
Blair	0.032084	0.326141	0.215822	1.000000	-0.243508	-0.295944	-0.021299
Hague	0.031144	-0.200790	-0.100392	-0.243508	1.000000	0.285738	-0.029906
Europe	0.064562	-0.209150	-0.112897	-0.295944	0.285738	1.000000	-0.151197
political.knowledge	-0.046598	-0.023510	-0.038528	-0.021299	-0.029906	-0.151197	1.000000

- Pair-plot with Hue on dependent variable “vote” to check if the independent variables are weak, poor or good predictors for dependent variable: -





- Plotting hear map for correlation of multiple variables: -



- Checking if dependent variable is balanced of not: -

vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
Conservative	460	460	460	460	460	460	460	460
Labour	1057	1057	1057	1057	1057	1057	1057	1057

INFERENCES FROM ABOVE UNIVARIATE, BIVARIATE AND MULTIVARIATE ANALYSIS

- ✓ From univariate analysis we can find that the variables “economic.cond.household” and “economic.cond.national” have outliers with percentage of 4.28 and 2.44 respectively, Remaining all other variable doesn’t show any outliers. However, treatment of these outliers is not required as these are categorical variables in nature.
- ✓ We can also conclude that the data is not scaled as we can see variations in standard deviation and other statistical parameters of the variables. However, scaling of data is not required as variables are categorical in nature.
- ✓ Dataset doesn’t have any null values.
- ✓ The variable “vote” and “gender” are of object datatypes and remaining variables are of numeric/int data types.
- ✓ From distribution plot and boxplot, we can see the data distribution for each variable.
- ✓ Dataset have 9 variables and 1525 observations. Dataset also have 8 duplicated observations deleting which resulted in total of 1517 observations.
- ✓ From multivariable analysis using pair-plot we can conclude that all variables are independent in nature and distribution of one variable doesn’t affect the flow of another variable.
- ✓ Upon checking co-variance and correlation of multiple variables confirms that all the variables are independent in nature.
- ✓ From pair-plot we can also conclude that the variables “age”, “economic.cond.national”, “economic.cond.household” and “political knowledge” are poor predictors for dependent variable.
- ✓ Other variables “Blair”, “Hague” and “Europe” are weak predictors for dependent variable.
- ✓ From heat map of correlation, we can see that none of the variables are showing strong correlation of more than 50%. Which indicates independency of variables.
- ✓ The data is imbalanced.
- ✓ From Bi-variate analysis we can conclude that the “gender” variable does not impact the votes for Labour and Conservative parties.
- ✓ The people with different political knowledge does not impact the votes among Labour and Conservative parties.
- ✓ From Bi-Variate analysis we can see that people with low attitude towards European integration votes majorly for Labour Party.
- ✓ From bi-variate analysis we can conclude that people with high assessment on current national economic condition (economic.cond.national), majority votes for Labour party.

PART-2 -- Data Preparation: -

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).

Solution: -

- From the above descriptive analysis we have noticed that the variables "vote" and "gender" are of object data types and to proceed with further analysis we need to convert them to numeric/categorical data type as python environment works best with numeric data types.
- Used LabelEncoder from sklearn.preprocessing to convert variables with object data types into categorical.
- Can use GetDummies function as well however LabelEncoder works best for variable with 2 categories without changing the variable name and without increasing number of variables. However, output of both the procedure will be the same.
- A view of dataset after label encoding: -

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	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

- With Label Encoding the dataset now takes "Labour" as "1" and "Conservative" as "0" for "vote" variable and "Female" as "0" and "Male" as "1" for "gender" variable.
- A view of datatypes of variables after label encoding and we can see that all are of integer data types.: -

```
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   vote                   1517 non-null   int32
 1   age                    1517 non-null   int64
 2   economic.cond.national 1517 non-null   int64
 3   economic.cond.household 1517 non-null  int64
 4   Blair                  1517 non-null   int64
 5   Hague                  1517 non-null   int64
 6   Europe                 1517 non-null   int64
 7   political.knowledge    1517 non-null   int64
 8   gender                 1517 non-null   int32
```

- Checking on standard deviation of variables in dataset: -

```
age                     15.701741
economic.cond.national   0.881792
economic.cond.household  0.931069
Blair                    1.174772
Hague                    1.232479
Europe                   3.299043
political.knowledge      1.084417
```

- From above we can see that the variables differ on their values for standard deviation and variance, with which we can conclude that the data is not scaled. However, in this given dataset scaling is not necessary as most of the independent variables are categorical in nature and it makes sense to perform scaling in pre-processing for variables continuous in nature but not for variables categorical in nature. Hence, Scaling is **NOT** necessary for this dataset.

- As instructed, we have divided data set into training and testing datasets in 70:30 ratio as defined.
- Below is the view of shape of training and testing dataset with corresponding labels: -

```
X_train (1061, 8)
X_test (456, 8)
y_train (1061,)
y_test (456,)
```

- After split the training dataset has 1061 observations and testing dataset has 456 observations.

PART-3 -- Modelling: -

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

AND

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

Solutions: -

BUILDING LOGISTIC REGRESSION MODEL: -

Building model with default hyperparameters: -

Post splitting data into training and testing data set we fitted logistic regression model into training dataset and performed prediction on training and testing dataset using the same model. We made the first model with default hyperparameters with default solver as Ibfgs.

Below are the accuracy scores obtained from this model: -

```
Accuracy of training dataset: 0.8350612629594723

Accuracy of testing dataset: 0.8245614035087719
```

Below is the confusion matrix obtained from this model: -

```
Confusion Matrix for train dataset

array([[199, 108],
       [ 67, 687]], dtype=int64)


Confusion Matrix for test dataset

array([[110,  43],
       [ 37, 266]], dtype=int64)
```

Below is the classification report obtained from this model: -

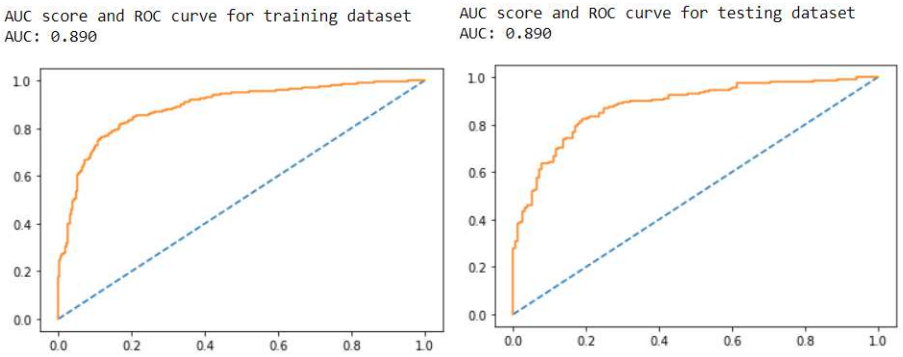
Classification report for train dataset

	precision	recall	f1-score	support
0	0.75	0.65	0.69	307
1	0.86	0.91	0.89	754
accuracy			0.84	1061
macro avg	0.81	0.78	0.79	1061
weighted avg	0.83	0.84	0.83	1061

Classification report for test dataset

	precision	recall	f1-score	support
0	0.75	0.72	0.73	153
1	0.86	0.88	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

Below is the AUC score and ROC curve obtained from this model: -



Below are the 10-fold cross validation for logistic regression with default values: -

Cross-validation is a process to check if the built model is correct or not.

Below are the 10-fold cross validation scores: -

```
cross validation scroes for traning dataset

array([0.82242991, 0.77358491, 0.83962264, 0.86792453, 0.85849057,
       0.8490566 , 0.81132075, 0.8490566 , 0.81132075, 0.83018868])

cross calidation scores for testing dataset

array([0.80434783, 0.7826087 , 0.84782609, 0.82608696, 0.89130435,
       0.86956522, 0.93333333, 0.84444444, 0.73333333, 0.82222222])
```

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Building model using GridSearchCV and analysing the best parameters: -

We use GridSearchCV to find an optimal combination of hyperparameters that minimizes a predefined loss function to give better results.

Performed GridSearchCV with various hyper-parameters like “solver”, “penalty” and “tol” and we can find that “linlinear” solver along with “l1” penalty worked as best parameters for this dataset. However, we have also observed that the difference in accuracy for train and test dataset is very marginal and not much of significant.

Below are the accuracy scores obtained from this model using GridSearchCV: -

```
Accuracy of training dataset after gridsearchCV: 0.8360037700282752

Accuracy of testing dataset after gridsearchCV: 0.8289473684210527
```

Below is the confusion matrix obtained from this model using GridSearchCV: -

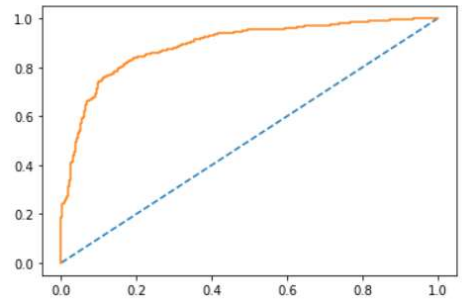
```
confusion matrix for train dataset

array([[196, 111],
       [ 63, 691]], dtype=int64)

confusion matrix for test dataset

array([[110,  43],
       [ 35, 268]], dtype=int64)
```

Below is the confusion matrix obtained from this model using GridSearchCV: -

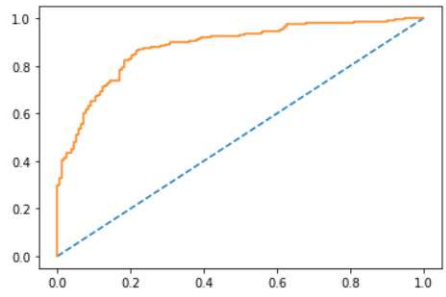
```
Classification report for train dataset
              precision    recall  f1-score   support

           0       0.76      0.64      0.69       307
           1       0.86      0.92      0.89       754

    accuracy                           0.84      1061
   macro avg       0.81      0.78      0.79      1061
weighted avg       0.83      0.84      0.83      1061


 Classification report for test dataset
              precision    recall  f1-score   support

           0       0.76      0.72      0.74       153
           1       0.86      0.88      0.87       303

    accuracy                           0.83       456
   macro avg       0.81      0.80      0.81       456
weighted avg       0.83      0.83      0.83       456
```

Below is the AUC score and ROC curve obtained from this model using GridSearchCV: -

AUC score and ROC curve for training dataset
AUC: 0.890



AUC score and ROC curve for testing dataset
AUC: 0.881



Below are the 10-fold cross validation scores: -

```
cross validation score for training dataset

array([0.8317757 , 0.78301887, 0.81132075, 0.82075472, 0.83962264,
       0.82075472, 0.77358491, 0.83962264, 0.79245283, 0.79245283])
```

```
cross calidation score for testing dataset

array([0.80434783, 0.7826087 , 0.82608696, 0.67391304, 0.86956522,
       0.73913043, 0.82222222, 0.82222222, 0.75555556, 0.71111111])
```

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Building model using SMOTE: -

In our previous analysis we have seen that the data is imbalanced in nature. We can use SMOTE technique to balance the data and then tried building model on the balanced data to check if we can see some significant improvement in accuracy for training and testing dataset. After building model on balanced dataset and checking on accuracy we can see that the performance is not that significant in terms of accuracy.

Below are the accuracy scores obtained from balanced data: -

```
Accuracy of training dataset: 0.8395225464190982

Accuracy of testing dataset: 0.7982456140350878
```

Below is the confusion matrix obtained from balanced data: -

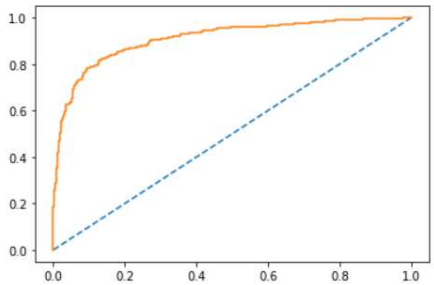
```
Confusion matrix for train dataset

array([[637, 117],
       [125, 629]], dtype=int64)

Confusion matrix for test dataset

array([[123,  30],
       [ 62, 241]], dtype=int64)
```

Below is the classification report obtained from balanced data: -

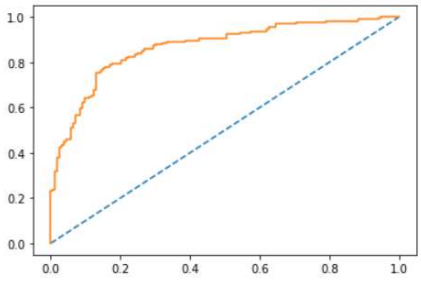
```
Classification report for train dataset
              precision    recall  f1-score   support

           0       0.84      0.84      0.84       754
           1       0.84      0.83      0.84       754

    accuracy                           0.84      1508
   macro avg       0.84      0.84      0.84      1508
weighted avg       0.84      0.84      0.84      1508

Classification report for test dataset
              precision    recall  f1-score   support

           0       0.66      0.80      0.73       153
           1       0.89      0.80      0.84       303

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

Below are the AUC scores and ROC curve obtained from balanced data: -

AUC score and ROC curve for training dataset
AUC: 0.910



AUC score and ROC curve for testing dataset
AUC: 0.867



Below are the 10-fold cross validation scores: -

```
cross validation score for balanced training dataset

array([0.76821192, 0.83443709, 0.82119205, 0.82119205, 0.90728477,
       0.81456954, 0.84768212, 0.86754967, 0.78666667, 0.89333333])
```

```
cross validation score for testing dataset

array([0.80434783, 0.7826087 , 0.84782609, 0.82608696, 0.89130435,
       0.86956522, 0.93333333, 0.84444444, 0.73333333, 0.82222222])
```

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Inference/Conclusion from Logistic Regression Model: -

From the above we can conclude that the data is **neither "Overfit" nor "Underfit"** in nature. And we can also inference that the model built using GridSearchCV is best optimized considering the best parameters obtained. However, the accuracy scores along with recall, precision, F1 values, ROC curve and AUC score are not that significant as compared with models built with default values and balanced data (SMOTE). Model built on balanced data set using SMOTE technique works well in training dataset however we can see a significant deplete in accuracy when it comes to testing dataset.

BUILDING LDA MODEL: -

Building model with default hyperparameters: -

From the above split into training and testing dataset we are building Linear Discriminant Analysis (LDA) model to check it this can outperform Logistic regression model and we can choose form best for further predictions. Firstly, we are building model using default values of LDA. That is, solver as "svd" and shrinkage as "none".

Below are the accuracy scores obtained from this model: -

```
Accuracy score of training dataset: 0.8341187558906692

Accuracy score of testing dataset: 0.8333333333333334
```

Below is the confusion matrix obtained from this model: -

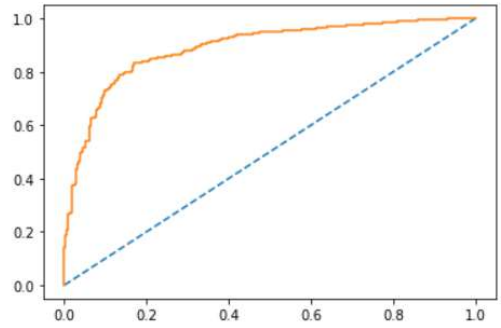
```
Confusion matrix of training dataset

array([[200, 107],
       [ 69, 685]], dtype=int64)


Confusion matrix of testing dataset

array([[111,  42],
       [ 34, 269]], dtype=int64)
```

Below is the classification report obtained from this model: -

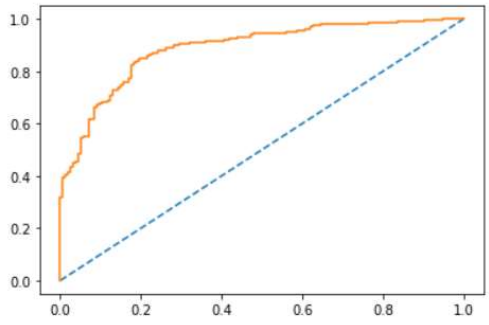
```
Classification Report of the training data:

              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


Classification Report of the test data:

              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
```

Below are the AUC scores and ROC curves obtained from this model: -

AUC score and ROC curve for training dataset
AUC: 0.867



AUC score and ROC curve for testing dataset
AUC: 0.867



Below are the 10-fold cross validation scores: -

```
cross validation score for training dataset

array([0.79439252, 0.77358491, 0.83962264, 0.85849057, 0.85849057,
       0.8490566 , 0.80188679, 0.8490566 , 0.81132075, 0.82075472])


cross validation score for testing dataset

array([0.80434783, 0.76086957, 0.86956522, 0.82608696, 0.89130435,
       0.86956522, 0.93333333, 0.84444444, 0.75555556, 0.84444444])
```

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Building model using GridSearchCV and analysing the best parameters: -

Using GridSearchCV function we tried finding the best parameters to further tune-in the above model for better accuracy and we have find that shrinkage as “auto”, solver as “lsqr” and tol value of “0.001” gives the best model considering accuracy, precision, recall, F1, ROC curve and AUC score.

Below are the accuracy scores obtained from model built using GridSearchCV: -

```
Accuracy of training dataset after gridsearchCV: 0.8331762488218661

Accuracy of testing dataset after gridsearchCV: 0.8355263157894737
```

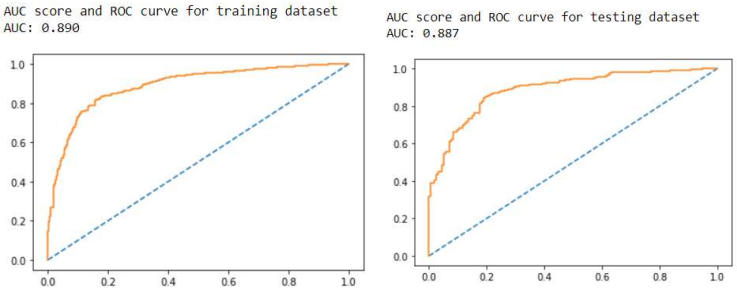
Below is the confusion matrix obtained from model built using GridSearchCV: -

```
confusuon matrix for training dataset

array([[200, 107],
       [ 70, 684]], dtype=int64)

confusuon matrix for testing dataset

array([[113,  40],
       [ 35, 268]], dtype=int64)
```

Below are the classification reports obtained from model built using GridSearchCV: -

```
Classification report for train dataset
              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

Classification report for test dataset
              precision    recall  f1-score   support

           0       0.76      0.74      0.75       153
           1       0.87      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
```

Below are the ROC curve and AUC scores obtained from model built using GridSearchCV: -



Below are the 10-fold cross validation scores: -

```
cross validation scores for training dataset

array([0.8317757 , 0.78301887, 0.81132075, 0.82075472, 0.83962264,
       0.82075472, 0.77358491, 0.83962264, 0.79245283, 0.79245283])


cross validation scores from testing dataset

array([0.80434783, 0.76086957, 0.86956522, 0.82608696, 0.89130435,
       0.86956522, 0.93333333, 0.84444444, 0.75555556, 0.84444444])
```

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Building model using SMOTE: -

From above descriptive analysis we can conclude that the original data provided is imbalance in nature and by using SMOTE technique we can balance the data to check if the model can outperform when data is balanced. We have applied SMOTE technique to oversample the data and to obtain a balanced dataset.

Below are the accuracy scores obtained from balanced dataset: -

```
Accuracy of training dataset: 0.8408488063660478

Accuracy of testing dataset: 0.8070175438596491
```

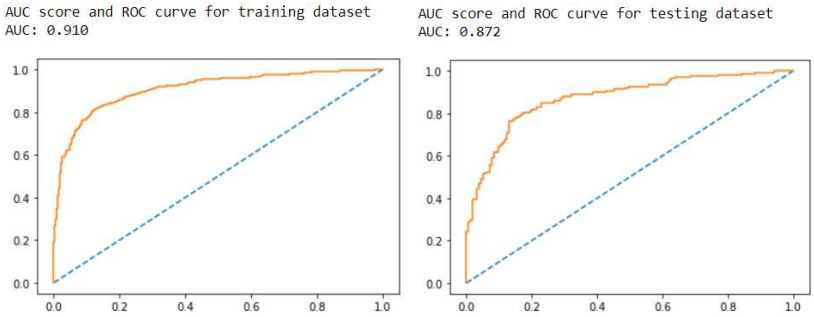
Below is the confusion matrix obtained from balanced dataset: -

```
confusion matrix for training dataset

array([[640, 114],
       [126, 628]], dtype=int64)

confusion matrix for testing dataset

array([[126,  27],
       [ 61, 242]], dtype=int64)
```

Below is the classification report obtained from balanced dataset: -

```
Classification report for train dataset
              precision    recall  f1-score   support

           0       0.84      0.85      0.84       754
           1       0.85      0.83      0.84       754

    accuracy                           0.84      1508
   macro avg       0.84      0.84      0.84      1508
weighted avg       0.84      0.84      0.84      1508

 Classification report for test dataset
              precision    recall  f1-score   support

           0       0.67      0.82      0.74       153
           1       0.90      0.80      0.85       303

    accuracy                           0.81       456
   macro avg       0.79      0.81      0.79       456
weighted avg       0.82      0.81      0.81       456
```

Below are the ROC curve and AUC scores obtained from balanced dataset: -



Below are the 10-fold cross validation scores: -

```
cross validation scores for training dataset

array([0.78145695, 0.84768212, 0.82781457, 0.82781457, 0.90728477,
       0.81456954, 0.82781457, 0.86754967, 0.78666667, 0.89333333])


cross validation scores for testing dataset

array([0.80434783, 0.76086957, 0.86956522, 0.82608696, 0.89130435,
       0.86956522, 0.93333333, 0.84444444, 0.75555556, 0.84444444])
```

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Inference/Conclusion from LDA Model: -

From the above we can conclude that the data is **neither "Overfit" nor "Underfit"** in nature. And we can also inference that the model built using the parameters from GridSearchCV is best optimized. However, the accuracy scores along with recall, precision, F1 values, ROC curve and AUC score are not that significant as compared with models built with default values and model built using GridSearchCV. Model built on balance data set using SMOTE technique works well in training dataset however we can see a significant deplete in accuracy when it comes to testing dataset.

BUILDING KNN MODEL: -

Building model with default hyperparameters: -

Post splitting data into training and testing data set we fitted KNN model into training dataset and performed prediction on training and testing dataset using the same model. We made the first model with default hyperparameters with default value of n_neighbour as "5".

Below are the accuracy scores obtained from this model: -

```
Accracy of training dataset: 0.8529688972667295

accuracy for testing dataset 0.8157894736842105
```

Below are the confusion matrices obtained from this model: -

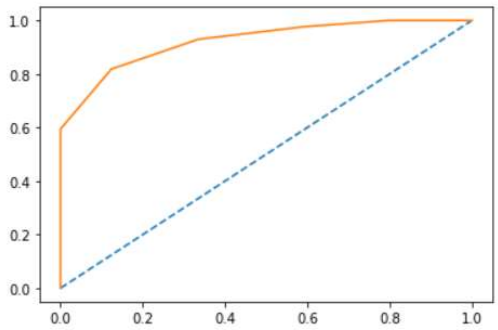
```
confusion matrix of training dataset
[[204 103]
 [ 53 701]]

confusion matrix for testing dataset
[[ 99  54]
 [ 30 273]]
```

Below are the classification reports obtained from this model: -

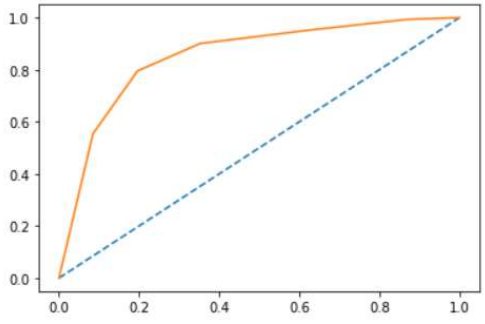
```
classificatoin report of training dataset
              precision    recall  f1-score   support

           0       0.79      0.66      0.72       307
           1       0.87      0.93      0.90       754

    accuracy                           0.85      1061
   macro avg       0.83      0.80      0.81      1061
weighted avg       0.85      0.85      0.85      1061


classsification report for testing dataset
              precision    recall  f1-score   support

           0       0.77      0.65      0.70       153
           1       0.83      0.90      0.87       303

    accuracy                           0.82       456
   macro avg       0.80      0.77      0.78       456
weighted avg       0.81      0.82      0.81       456
```

Below are the AUC scores and ROC curves obtained from this model: -

AUC score and ROC curve for training dataset
AUC: 0.923



AUC score and ROC curve for testing dataset
AUC: 0.852



Below are the 10-fold cross validation scores: -

```
cross validation scores for train dataset

array([0.81308411, 0.77358491, 0.78301887, 0.77358491, 0.80188679,
       0.83018868, 0.83018868, 0.83962264, 0.76415094, 0.80188679])


cross validation scores for test dataset

array([0.67391304, 0.73913043, 0.82608696, 0.65217391, 0.7826087 ,
       0.69565217, 0.82222222, 0.73333333, 0.66666667, 0.73333333])
```

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

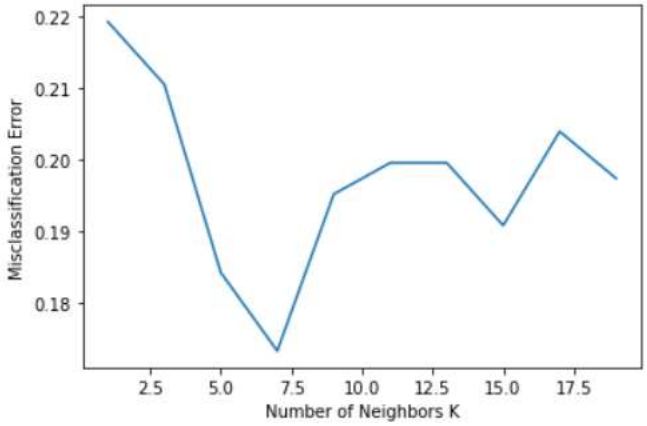
Find the right value of n_neighbor: -

Its very important to have the right value of n_neighbors to fetch the best accuracy from the model. We can decide on the best value for n_neighbors based on MSE (mean squared error) scores. The value with least score of MSE indicated least error and will fetch the best optimized n_neoghbors value.

Below are the MSE scores: -

```
[0.2192982456140351,
 0.21052631578947367,
 0.1842105263157895,
 0.17324561403508776,
 0.19517543859649122,
 0.19956140350877194,
 0.19956140350877194,
 0.1907894736842105,
 0.20394736842105265,
 0.19736842105263153]
```

Below is the graphical version of of MSE scores acorss numerous values of n_neighbors.



From the above plotted graph we can see the n_neighbors with value "7" gives the least MSE score. With which we can proceed and build KNN model with n_neighbor value as "7" to check if this model and outperform when compared to model built with default n_neighbors value as "5".

Below are the accuracy scored with n_neighbors values as "7": -

```
accuracy for training dataset: 0.8444863336475024

accuracy score for testing dataset: 0.8267543859649122
```

Below are the confusion matrices with n_neighbors values as "7": -

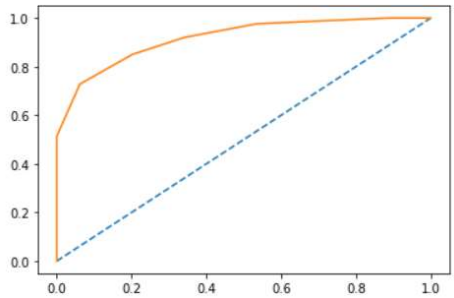
```
confusion matrix for training dataset
[[171 136]
 [ 50 704]]

confusion matrix for testing dataset
[[100  53]
 [ 26 277]]
```

Below are the classification reports with n_neighbors values as "7": -

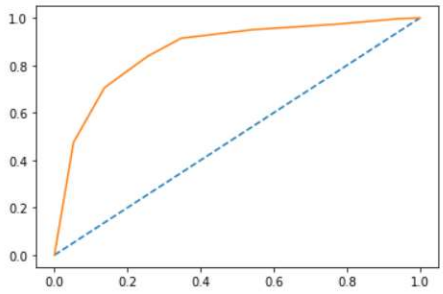
```
classification report for training dataset
              precision    recall  f1-score   support

           0       0.77      0.56      0.65       307
           1       0.84      0.93      0.88       754

    accuracy                           0.82      1061
   macro avg       0.81      0.75      0.77      1061
weighted avg       0.82      0.82      0.82      1061


classification report for testing dataset
              precision    recall  f1-score   support

           0       0.79      0.65      0.72       153
           1       0.84      0.91      0.88       303

    accuracy                           0.83       456
   macro avg       0.82      0.78      0.80       456
weighted avg       0.82      0.83      0.82       456
```

Below are the AUC scores and ROC curves with n_neighbors values as “7”: -

AUC score and ROC curve for training dataset
AUC: 0.917



AUC score and ROC curve for testing dataset
AUC: 0.863



Below are the 10-fold cross validation scores: -

```
cross validation scores for training dataset

array([0.80373832, 0.78301887, 0.79245283, 0.81132075, 0.83962264,
       0.83962264, 0.78301887, 0.85849057, 0.76415094, 0.75471698])


cross validation scores for testing dataset

array([0.76086957, 0.82608696, 0.82608696, 0.69565217, 0.89130435,
       0.80434783, 0.88888889, 0.68888889, 0.73333333, 0.71111111])
```

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Building model using GridSearchCV and getting the best hyperparameters: -

After building the model with its default values as shown above, we will try and find the best hyper parameters to check if we can outperform the accuracy achieved by the model built with default values of hyperparameter. From GridSearchCV we found that the best parameters are “auto” as algorithm, “Manhattan” as metrics, “9” as n_neighbors and “uniform” as weights.

Below are the accuracy scores obtained from this model using GridSearchCV: -

```
Accuracy of training dataset after gridsearchCV: 0.8520263901979265

Accuracy of testing dataset after gridsearchCV: 0.8026315789473685
```

Below are the confusion matrices obtained from this model using GridSearchCV: -

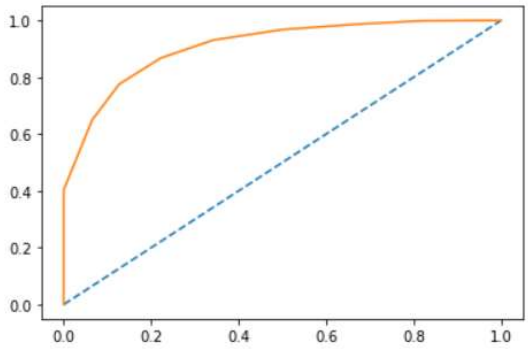
```
confusuon matrix for training dataset

array([[202, 105],
       [ 52, 702]], dtype=int64)


confusuon matrix for testing dataset

array([[ 95,  58],
       [ 32, 271]], dtype=int64)
```

Below are the classification reports obtained from this model using GridSearchCV: -

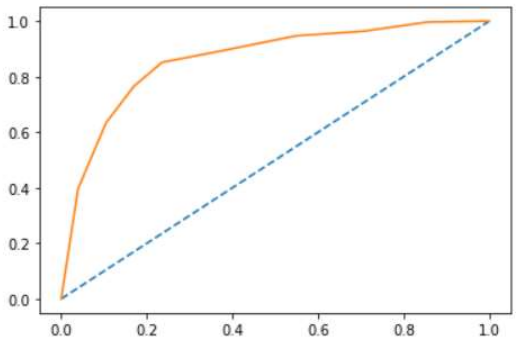
```
Classification report for train dataset
              precision    recall  f1-score   support

           0       0.80      0.66      0.72       307
           1       0.87      0.93      0.90       754

    accuracy                           0.85      1061
   macro avg       0.83      0.79      0.81      1061
weighted avg       0.85      0.85      0.85      1061


Classification report for test dataset
              precision    recall  f1-score   support

           0       0.75      0.62      0.68       153
           1       0.82      0.89      0.86       303

    accuracy                           0.80       456
   macro avg       0.79      0.76      0.77       456
weighted avg       0.80      0.80      0.80       456
```

Below are the AUC scores and ROC curves obtained from this model using GridSearchCV: -

AUC score and ROC curve for training dataset
AUC: 0.909



AUC score and ROC curve for testing dataset
AUC: 0.863



Below are the 10-fold cross validation scores: -

```
cross validation scores for train dataset

array([0.8317757 , 0.78301887, 0.81132075, 0.82075472, 0.83962264,
       0.82075472, 0.77358491, 0.83962264, 0.79245283, 0.79245283])


cross validation scores for test dataset

array([0.80434783, 0.7826087 , 0.82608696, 0.67391304, 0.86956522,
       0.73913043, 0.82222222, 0.82222222, 0.75555556, 0.71111111])
```

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Building model using SMOTE: -

From above descriptive analysis we can conclude that the original data provided is imbalance in nature and by using SMOTE technique we can balance the data to check if the model can outperform when data is balanced. We have applied SMOTE technique to oversample the data and to obtain a balanced dataset.

Below are the accuracy scores obtained from balanced dataset: -

```
Accuracy of training dataset: 0.8362068965517241

Accuracy of testing dataset: 0.7785087719298246
```

Below are the confusion matrices obtained from balanced dataset: -

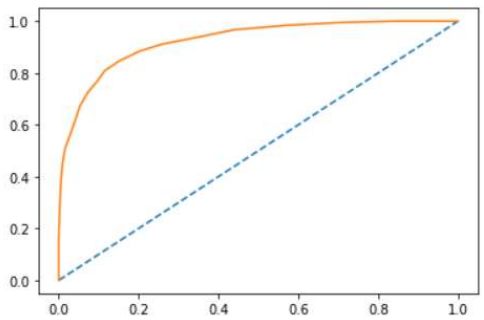
```
confusion matrix for training dataset

array([[681,  73],
       [174, 580]], dtype=int64)

confusion matrix for testing dataset

array([[128,  25],
       [ 76, 227]], dtype=int64)
```

Below are the classification reports obtained from balanced dataset: -

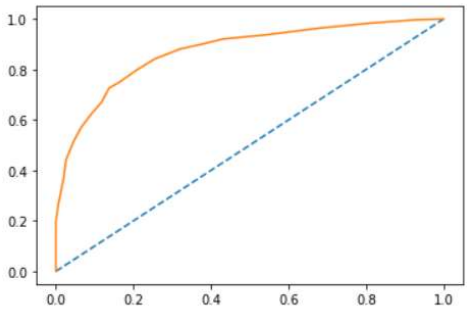
```
Classification report for train dataset
              precision    recall  f1-score   support

           0       0.80      0.90      0.85       754
           1       0.89      0.77      0.82       754

    accuracy                           0.84      1508
   macro avg       0.84      0.84      0.84      1508
weighted avg       0.84      0.84      0.84      1508

Classification report for test dataset
              precision    recall  f1-score   support

           0       0.63      0.84      0.72       153
           1       0.90      0.75      0.82       303

    accuracy                           0.78       456
   macro avg       0.76      0.79      0.77       456
weighted avg       0.81      0.78      0.78       456
```

Below are the AUC scores and ROC curves obtained from balanced dataset: -

AUC score and ROC curve for training dataset
AUC: 0.924



AUC score and ROC curve for testing dataset
AUC: 0.872



Below are the 10-fold cross validation scores: -

```
cross validation scores for train dataset

array([0.78807947, 0.78807947, 0.77483444, 0.7615894 , 0.84768212,
       0.81456954, 0.79470199, 0.8013245 , 0.68      , 0.84666667])


cross validation scores for test dataset

array([0.67391304, 0.73913043, 0.82608696, 0.65217391, 0.7826087 ,
       0.69565217, 0.82222222, 0.73333333, 0.66666667, 0.73333333])
```

we can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Inference/Conclusion from KNN Model: -

From the above we can conclude that the data is **neither "Overfit" nor "Underfit"** in nature. And we can also inference that the model built using n_neighbors value as "7" is best optimized model for prediction. However, we can see significant variations in accuracy score, F1 score, recall values, precision values, ROC curves and AUC scores when compared with default values of KNN and also with model built on balanced dataset. Model built on balance data set using SMOTE technique works well in training dataset however we can see a significant deplete in accuracy when it comes to testing dataset.

BUILDING NAÏVE BAYES MODEL: -

Post splitting data into training and testing we are now ready to build model using Naïve Bayes algorithm. Naïve bayes algorithm is based out of Bayes theorem of conditional probability. considering that all events are independent of each other and then we find the probability of an event happening under the condition that one of the events has already occurred.

Below are accuracy scores using this model: -

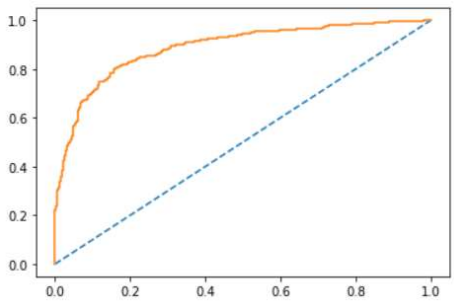
```
Accracy of training dataset: 0.8350612629594723

Accracy of testing dataset: 0.8223684210526315
```

Below are confusion matrices and classification reports using this model: -

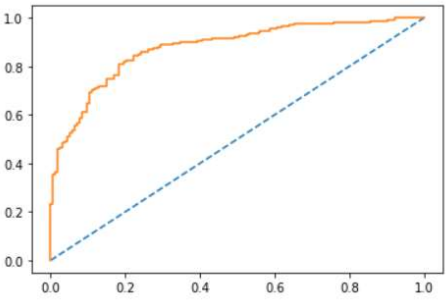
```
Confusion matrix of train dataset
[[211  96]
 [ 79 675]]
              precision    recall  f1-score   support

           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


Confusion matrix of test dataset
[[112  41]
 [ 40 263]]
Classification report of test dataset
              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
```

Below are AUC scores and ROC curves using this model: -

AUC score and ROC curve for training dataset
AUC: 0.888



AUC score and ROC curve for testing dataset
AUC: 0.876



Below are 10-fold cross validation scores using this model: -

```
cross validation scores for train dataset

array([0.80373832, 0.78301887, 0.8490566 , 0.83962264, 0.90566038,
       0.8490566 , 0.78301887, 0.83962264, 0.81132075, 0.82075472])

cross validation scores for test dataset

array([0.7826087 , 0.80434783, 0.86956522, 0.80434783, 0.86956522,
       0.86956522, 0.88888889, 0.82222222, 0.75555556, 0.82222222])
```

Building Gaussian Naive Bayes over balanced data using SMOTE

After building naïve bayes model using original imbalanced data. Now, we can try and build the same model using balanced data to check if it outperforms in terms of accuracy and other measurement factors.

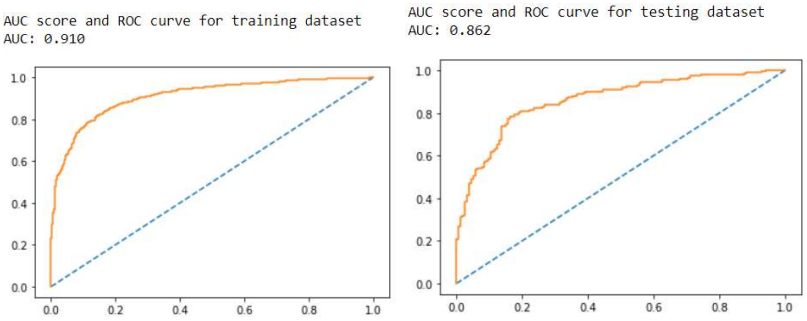
Below are the accuracy scores obtained using Naïve Bayes algorithm over balanced dataset: -

```
Accracy of training dataset: 0.8355437665782494

Accracy of testing dataset: 0.7982456140350878
```

Below are the confusion matrices and classification reports obtained using Naïve Bayes algorithm over balanced dataset: -

```
Confusion matrix of train dataset
[[631 123]
 [125 629]]
              precision    recall  f1-score   support

           0       0.83      0.84      0.84       754
           1       0.84      0.83      0.84       754

    accuracy                           0.84      1508
   macro avg       0.84      0.84      0.84      1508
weighted avg       0.84      0.84      0.84      1508

Confusion matrix of test dataset
[[118  35]
 [ 57 246]]
Classification report of test dataset
              precision    recall  f1-score   support

           0       0.67      0.77      0.72       153
           1       0.88      0.81      0.84       303

    accuracy                           0.80       456
   macro avg       0.77      0.79      0.78       456
weighted avg       0.81      0.80      0.80       456
```

Below are the AUC scores and ROC curves obtained using Naïve Bayes algorithm over balanced dataset: -



Below are the 10-fold cross validation scores obtained using Naïve Bayes algorithm over balanced dataset: -

```
cross validation scores for train dataset

array([0.76821192, 0.82119205, 0.82781457, 0.77483444, 0.91390728,
       0.82781457, 0.82781457, 0.86092715, 0.8       , 0.87333333])

cross validation scores for test dataset

array([0.7826087 , 0.80434783, 0.86956522, 0.80434783, 0.86956522,
       0.86956522, 0.88888889, 0.82222222, 0.75555556, 0.82222222])
```

Inference/Conclusion from Naïve Bayes Model: -

From the above we can conclude that the data is **neither "Overfit" nor "Underfit"** in nature. And we can also inference that the model built using original imbalanced data is best optimized model for prediction. However, we can see significant variations in accuracy score, F1 score, recall values, precision values, ROC curves and AUC scores when compared with model built on balanced dataset. Model built on balance data set using SMOTE technique works well in training dataset however we can see a significant deplete in accuracy when it comes to training dataset.

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

Solution: -

Model Tuning: -

We have performed model tuning for all the models built that is Logistic regression, LDA and KNN using GridSearchCV. It is a function allows us to input all the probabilities of a hyper-parameter of a model and it provides us the best combination of hyper-parameters with minimized Log loss and on that combination, we can build a model to check if it outperforms the model built on default values of hyper-parameter.

For Naïve Bayes model we do not have any hyper-parameter as the algorithm by its nature works on bayes theorem of conditional probability and doesn't have distance measurement or metric as hyper-parameter. However, we have tried building Naïve bayes model with original data and balanced data.

Below are inferences from using GridSearchCV for tuning model: -

- For Logistic regression, model built using GridSearchCV has performed little better as compared with default values of hyper-parameter. However. Model built with balanced dataset worked well with training data however the accuracy depleted when it comes to training data.
- For LDA, model built using best prams got from GridSearchCV is best optimized considering the accuracy scores for training and testing dataset. However, the accuracy values are not that significant but a little better than default values if Hyper-parameter. Also, Model built with balanced dataset worked well with training data however the accuracy depleted when it comes to training data.
- For KNN, Model built with "7" as n_neighbors are best optimized considering the accuracy scores along with all other parameter as default. Model built using best params from GridSearchCV performed little better on training dataset however we can notice depreciation in accuracy when it comes to testing dataset. Also, Model built with balanced dataset worked well with training data however the accuracy depleted when it comes to training data.
- For Naïve Bayes, the model built with original data (imbalanced) is best optimized considering accuracy, F1 score, precision and recall values. Model built with balanced dataset performed almost same on training dataset however the performance depreciated when it comes to testing dataset.

Pl. note - Model building is an iterative process. Model performance both on the test and train dataset can be improved using feature engineering, feature extraction, hyper parameter tuning (including combination of various parameters). Model has to match the business objective and hence various permutation and combinations can be tried on to refine the model.

BAGGING: -

Building Random Forest model: -

Firstly, let's build and random forest model for original dataset and balanced dataset and check the performance for the same.

Random Forest Built in original dataset: -

Below are the accuracy scores, confusion matrix and classification report for training dataset: -

```
accuracy score for training dataset: 1.0
confusion matrix for training dataset
[[307   0]
 [  0 754]]
classification report for training dataset
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       307
           1       1.00      1.00      1.00       754

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

Below are the accuracy scores, confusion matrix and classification report for testing dataset: -

```
accuracy score for testing dataset: 0.831140350877193
confusion matrix for testing dataset
[[104  49]
 [ 28 275]]
classification report for testing dataste
              precision    recall  f1-score   support

           0       0.79      0.68      0.73       153
           1       0.85      0.91      0.88       303

    accuracy                           0.83       456
   macro avg       0.82      0.79      0.80       456
weighted avg       0.83      0.83      0.83       456
```

Random Forest Built in balanced dataset: -

Below are the accuracy scores, confusion matrix and classification report for training dataset: -

```
accuracy score for training dataset: 1.0
confusion matrix for training dataset
[[754   0]
 [  0 754]]
classification report for training dataset
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       754
           1       1.00      1.00      1.00       754

    accuracy                           1.00      1508
   macro avg       1.00      1.00      1.00      1508
weighted avg       1.00      1.00      1.00      1508
```

Below are the accuracy scores, confusion matrix and classification report for testing dataset: -

```
accuracy score for testing dataset: 0.8267543859649122
confusion matrix for testing dataset
[[113  40]
 [ 39 264]]
classification report for testing dataste
              precision    recall  f1-score   support

           0       0.74      0.74      0.74       153
           1       0.87      0.87      0.87       303

    accuracy                           0.83       456
   macro avg       0.81      0.80      0.81       456
weighted avg       0.83      0.83      0.83       456
```

From the above random forest model built on original dataset and balanced dataset we can conclude that the model is overfitting as it gives 100% accuracy with training dataset however the accuracy depreciates when it comes to testing dataset. We have not create ROC curve and AUC scores as we know that the model is overfit.

Bagging on original dataset: -

Let's build bagging model using random forest and check if it can perform better than general random forest model.

Below are the accuracy scores, confusion matrix and classification report for training dataset: -

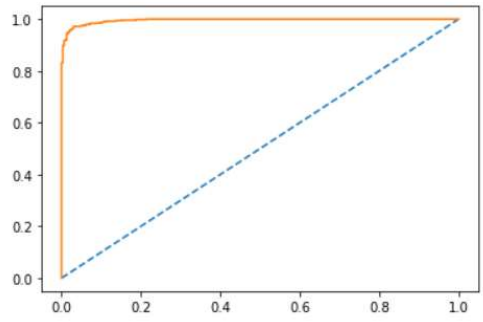
```
accuracy score or training dataset: 0.9613572101790764
confusion report for training dataset
[[274  33]
 [  8 746]]
classification report for training dataset
              precision    recall  f1-score   support

           0       0.97      0.89      0.93       307
           1       0.96      0.99      0.97       754

    accuracy                           0.96      1061
   macro avg       0.96      0.94      0.95      1061
weighted avg       0.96      0.96      0.96      1061
```

Below are the accuracy scores, confusion matrix and classification report for testing dataset: -

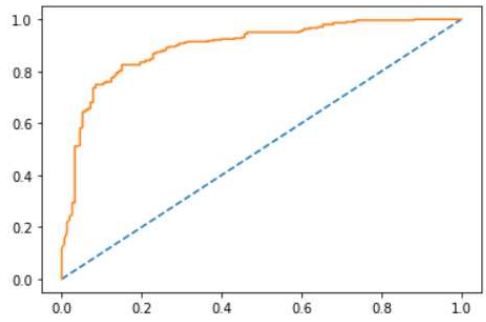
```
Accuracy score for testing datatset: 0.831140350877193
confusuion matrix for testing dataset
[[102  51]
 [ 26 277]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.80      0.67      0.73       153
           1       0.84      0.91      0.88       303

    accuracy                           0.83       456
   macro avg       0.82      0.79      0.80       456
weighted avg       0.83      0.83      0.83       456
```

Below is the AUC score and ROC curve for training and testing dataset: -

AUC score and ROC curve for training dataset
AUC: 0.996



AUC score and ROC curve for testing dataset
AUC: 0.895



- Inference from bagging: - Model is still overfitting as the difference in accuracy for training and testing dataset is more.

Bagging on Balanced dataset: -

Below are the accuracy scores, confusion matrix and classification report for training dataset:-

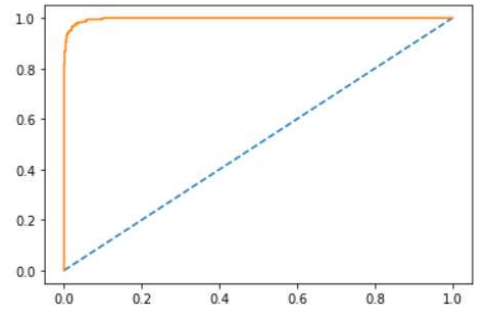
```
accuracy score or training dataset: 0.9721485411140584
confusion report for training dataset
[[739  15]
 [ 27 727]]
classification report for training dataset
              precision    recall  f1-score   support

           0       0.96      0.98      0.97       754
           1       0.98      0.96      0.97       754

    accuracy                           0.97      1508
   macro avg       0.97      0.97      0.97      1508
weighted avg       0.97      0.97      0.97      1508
```

Below are the accuracy scores, confusion matrix and classification report for testing dataset:-

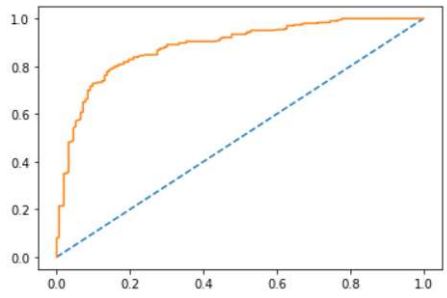
```
Accuracy score for testing datatset: 0.8114035087719298
confusuion matrix for testing dataset
[[111  42]
 [ 44 259]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.72      0.73      0.72       153
           1       0.86      0.85      0.86       303

    accuracy                           0.81       456
   macro avg       0.79      0.79      0.79       456
weighted avg       0.81      0.81      0.81       456
```

Below are the AUC scores and ROC curves for training and testing dataset: -

AUC score and ROC curve for training dataset
AUC: 0.997



AUC score and ROC curve for testing dataset
AUC: 0.884



Building Ada-Boost Model on original dataset: -

Below are the accuracy scores, confusion matrix and classification reports for training dataset: -

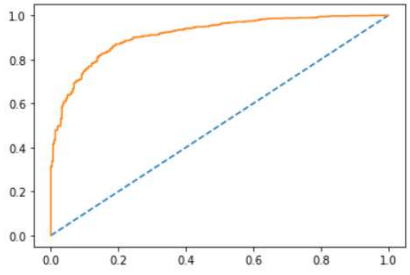
```
Accuracy for training dataset: 0.8463713477851084
confusion matrix for training dataset
[[210  97]
 [ 66 688]]
classification report for training dataset
              precision    recall  f1-score   support

           0       0.76      0.68      0.72       307
           1       0.88      0.91      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
```

Below are the accuracy scores, confusion matrix and classification reports for testing dataset: -

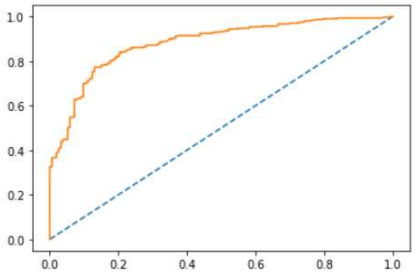
```
accuracy score for testing dataset: 0.8135964912280702
confusion matrix for testing dataset
[[105  48]
 [ 37 266]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.74      0.69      0.71       153
           1       0.85      0.88      0.86       303

    accuracy                           0.81       456
   macro avg       0.79      0.78      0.79       456
weighted avg       0.81      0.81      0.81       456
```

Below are the AUC scores and ROC curve for training and testing dataset: -

AUC score and ROC curve for training dataset
AUC: 0.912



AUC score and ROC curve for testing dataset
AUC: 0.881



Building Ada-Boost Model on balanced dataset: -

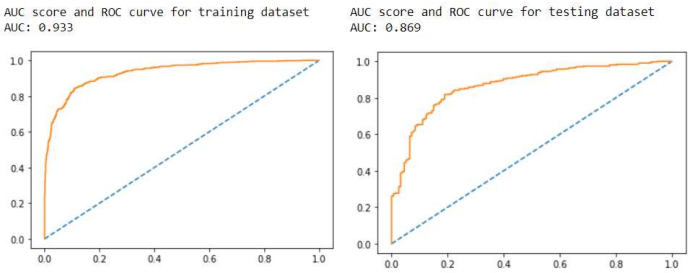
Below are the accuracy scores, confusion matrix and classification reports for training dataset: -

```
Accuracy for training dataset:
confusion matrix for training dataset
[[660  94]
 [110 644]]
classification report for training dataset
              precision    recall  f1-score   support

           0       0.86      0.88      0.87       754
           1       0.87      0.85      0.86       754

    accuracy                           0.86      1508
   macro avg       0.86      0.86      0.86      1508
weighted avg       0.86      0.86      0.86      1508
```

Below are the accuracy scores, confusion matrix and classification reports for testing dataset: -

```
accuracy score for testing dataset: 0.8201754385964912
confusion matrix for testing dataset
[[119  34]
 [ 48 255]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.71      0.78      0.74       153
           1       0.88      0.84      0.86       303

    accuracy                           0.82       456
   macro avg       0.80      0.81      0.80       456
weighted avg       0.83      0.82      0.82       456
```

Below are the AUC scores and ROC curve for training and testing dataset: -



Building Gradient Boosting Model on original dataset: -

Below are the accuracy scores, confusion matrix and classification reports for training dataset: -

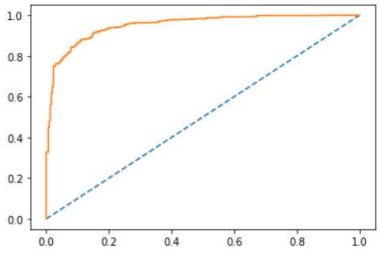
```
accuracy for training dataset: 0.8925541941564562
confusion matrix for training dataset
[[239  68]
 [ 46 708]]
classification report for training dataset
              precision    recall  f1-score   support

           0       0.84      0.78      0.81       307
           1       0.91      0.94      0.93       754

    accuracy                           0.89      1061
   macro avg       0.88      0.86      0.87      1061
weighted avg       0.89      0.89      0.89      1061
```

Below are the accuracy scores, confusion matrix and classification reports for testing dataset: -

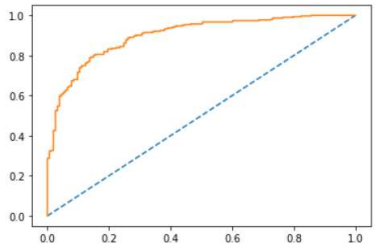
```
accuracy score for testing dataset: 0.8355263157894737
confusuon matrix for testing dataset
[[105  48]
 [ 27 276]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.80      0.69      0.74       153
           1       0.85      0.91      0.88       303

    accuracy                           0.84       456
   macro avg       0.82      0.80      0.81       456
weighted avg       0.83      0.84      0.83       456
```

Below are the AUC scores and ROC curve for training and testing dataset: -

AUC score and ROC curve for training dataset
AUC: 0.951



AUC score and ROC curve for testing dataset
AUC: 0.899



Building Gradient Boosting Model on balanced dataset: -

Below are the accuracy scores, confusion matrix and classification reports for training dataset: -

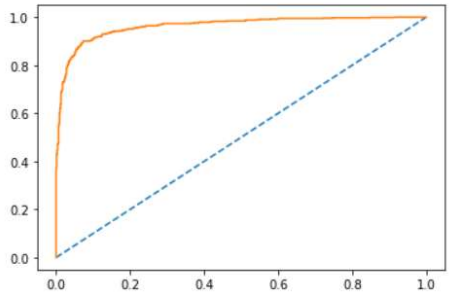
```
accuracy for training dataset: 0.9111405835543767
confusion matrix for training dataset
[[702  52]
 [ 82 672]]
classification report for training dataset
              precision    recall  f1-score   support

           0       0.90      0.93      0.91       754
           1       0.93      0.89      0.91       754

    accuracy                           0.91      1508
   macro avg       0.91      0.91      0.91      1508
weighted avg       0.91      0.91      0.91      1508
```

Below are the accuracy scores, confusion matrix and classification reports for testing dataset: -

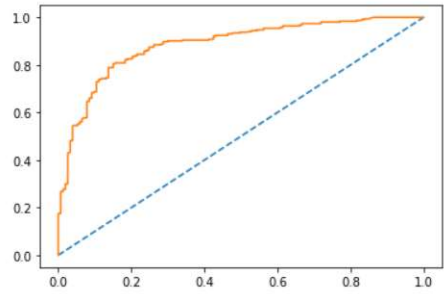
```
accuracy score for testing dataset: 0.8223684210526315
confusuon matrix for testing dataset
[[120  33]
 [ 48 255]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.71      0.78      0.75       153
           1       0.89      0.84      0.86       303

    accuracy                           0.82       456
   macro avg       0.80      0.81      0.81       456
weighted avg       0.83      0.82      0.82       456
```

Below are the AUC scores and ROC curve for training and testing dataset: -

AUC score and ROC curve for training dataset
AUC: 0.963



AUC score and ROC curve for testing dataset
AUC: 0.885



Inferences from Bagging and Boosting model: -

- Form both the model above we can notice that the model is still outfitted.
- When to comes to model performance over balanced dataset it performs well in training dataset however the accuracy reduces when it comes to training dataset.

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.

Solution: - Below is the table showing all the scores across models.

	Training dataset									Testing Dataset							
	Accuracy	Precision-0	Recall-0	F1 Score-0	Precision-1	Recall-1	F1 Score-1	AUC score		Accuracy	Precision-0	Recall-0	F1 Score-0	Precision-1	Recall-1	F1 Score-1	AUC score
Logistic Regression	83.51	0.75	0.65	0.69	0.86	0.91	0.89	0.89		82.46	0.75	0.72	0.73	0.86	0.88	0.87	0.89
Logistic Regression- CV	83.6	0.76	0.64	0.69	0.86	0.92	0.89	0.89		82.89	0.76	0.72	0.74	0.86	0.88	0.87	0.88
Logistic Regression- SM	83.95	0.84	0.84	0.84	0.84	0.83	0.84	0.91		79.82	0.66	0.8	0.73	0.89	0.8	0.84	0.87
LDA	83.41	0.74	0.65	0.69	0.86	0.91	0.89	0.87		83.33	0.77	0.73	0.74	0.86	0.89	0.88	0.87
LDA - CV	83.31	0.74	0.65	0.69	0.86	0.91	0.89	0.89		83.55	0.76	0.74	0.75	0.87	0.88	0.88	0.88
LDA - SM	84.08	0.84	0.85	0.84	0.85	0.83	0.84	0.91		80.71	0.67	0.82	0.74	0.91	0.81	0.85	0.87
KNN	85.29	0.79	0.66	0.72	0.87	0.93	0.9	0.92		81.57	0.77	0.65	0.71	0.83	0.9	0.87	0.85
KNN - 7	84.44	0.77	0.66	0.71	0.87	0.92	0.89	0.92		82.67	0.79	0.65	0.72	0.84	0.91	0.88	0.86
KNN - CV	85.2	0.8	0.66	0.72	0.87	0.93	0.9	0.91		80.26	0.75	0.62	0.68	0.82	0.89	0.86	0.86
KNN- SM	80.37	0.76	0.88	0.82	0.86	0.72	0.79	89.6		75.21	0.59	0.85	0.7	0.9	0.7	0.79	86.2
Naïve Bayes	83.51	0.73	0.69	0.71	0.88	0.9	0.89	0.88		82.23	0.74	0.73	0.73	0.87	0.87	0.87	0.88
Naïve Bayes -- SM	83.55	0.83	0.84	0.84	0.84	0.83	0.84	0.91		79.82	0.67	0.77	0.72	0.88	0.81	0.84	0.86
Bagging	96.13	0.97	0.89	0.93	0.96	0.99	0.97	0.99		83.11	0.8	0.67	0.73	0.84	0.91	0.88	0.89
Baggin - SM	97.2	0.96	0.98	0.97	0.98	0.96	0.97	0.99		81.11	0.72	0.73	0.72	0.86	0.85	0.86	0.88
Ada-Boosting	84.63	0.76	0.68	0.72	0.88	0.91	0.89	0.91		81.35	0.74	0.69	0.71	0.85	0.88	0.86	0.88
Ada-Boosting - SM	86.47	0.86	0.88	0.87	0.87	0.85	0.86	0.93		82.01	0.71	0.78	0.74	0.88	0.84	0.86	0.86
Gradient Boosting	89.25	0.84	0.78	0.81	0.91	0.94	0.93	0.95		83.55	0.8	0.69	0.74	0.85	0.91	0.88	0.89
Gradient Boosting - SM	91.11	0.9	0.93	0.91	0.93	0.89	0.91	0.96		82.23	0.71	0.78	0.75	0.89	0.85	0.86	0.88

Indicators/symbols for above tabular data: -

-CV : - indicates scores for model built on best params obtained from GridSearchCV with model name as prefix.

-SM: - indicates scores for model built on balanced dataset with model name as prefix.

KNN-7: - Indicates KNN model built with N_neighbors as "7".

Inferences on final model: -

- From the above tabular representation of all the scores for training and testing dataset across various model we can conclude that the LDA model with default values of hyper-parameters is best optimized for the given dataset. (highlighted in yellow)
- There is marginal difference in accuracy for Logistic regression and LDA, but comparatively LDA had a little better performance than logistic regression.
- In KNN with number of nearest neighbor's as "7" is also well optimized but difference in accuracy for training and testing dataset is little on the higher side as compared to LDA.
- Other model's namely Naïve Bayes, Bagging, Ada-Boost and Gradient boosting worked well on training dataset but the accuracy came down when performed over testing dataset. Which indicates overfitting of data in that model.
- All models built on balances dataset showed overfitting.
- We also understand that the accuracy and other measuring parameter of a model can be improved by trying various other combinations of hyper-parameter. Model building is an iterative process. Model performance both on training and testing dataset can be improves

using feature engineering, feature extraction and hyper-parameter tuning. Model has to match the business objective and hence various other combinations can be tried on to refine the model.

PART-4 -- Inferences:

1.8 Based on these predictions, what are the insights?

Solution: -

- Below are the insights found from all the above analysis: -
  - Data is imbalanced and may need more samples to have better understanding and analysis of data.
  - Almost 40% of people are not having adequate political knowledge.
  - Around 50% pollution is not convinced with current household economic condition.
  - Around 60% population is not satisfied with current national economic condition.
  - Assessment of Labour leader is more than leader from conservative party.

- Below are the recommendations derived from all the above analysis: -
  - If a party wants to win, they need to work on improving household economic condition and people need to be made aware of that.
  - If a party wants to win, they need to work around in improving national economic condition and awareness to be spread among people.
  - People need to be educated with political knowledge along with their roles and responsibilities to improve the current household and nation economic situation.
  - People need to be aware of the European integration and a survey can be conducted to analyse what they liked about integration and what they didn't liked about it.
  - Both the parties can promote the good deeds and future plans of their leader so that the populations understand the work and future ideas of these leaders.

----------END OF PROBLEM—1 ----------

PROBLEM—2

Objective: - 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 for analysis:

- President Franklin D. Roosevelt in 1941
- President John F. Kennedy in 1961
- President Richard Nixon in 1973

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

Solution: -

- We can see that there are 58 text files under inaugural corpus. Below are the details

```
['1789-Washington.txt',
 '1793-Washington.txt',
 '1797-Adams.txt',
 '1801-Jefferson.txt',
 '1805-Jefferson.txt',
 '1809-Madison.txt',
 '1813-Madison.txt',
 '1817-Monroe.txt',
 '1821-Monroe.txt',
 '1825-Adams.txt',
 '1829-Jackson.txt',
 '1833-Jackson.txt',
 '1837-VanBuren.txt',
 '1841-Harrison.txt',
 '1845-Polk.txt',
 '1849-Taylor.txt',
 '1853-Pierce.txt',
 '1857-Buchanan.txt',
 '1861-Lincoln.txt',
 '1865-Lincoln.txt',
 '1869-Grant.txt',
 '1873-Grant.txt',
 '1877-Hayes.txt',
 '1881-Garfield.txt',
 '1885-Cleveland.txt',
 '1889-Harrison.txt',
 '1893-Cleveland.txt',
 '1897-McKinley.txt',
 '1901-McKinley.txt',
 '1905-Roosevelt.txt',
 '1909-Taft.txt',
 '1913-Wilson.txt',
 '1917-Wilson.txt',
 '1921-Harding.txt',
 '1925-Coolidge.txt',
 '1929-Hoover.txt',
 '1933-Roosevelt.txt',
 '1937-Roosevelt.txt',
 '1941-Roosevelt.txt',
 '1945-Roosevelt.txt',
 '1949-Truman.txt',
 '1953-Eisenhower.txt',
 '1957-Eisenhower.txt',
 '1961-Kennedy.txt',
 '1965-Johnson.txt',
 '1969-Nixon.txt',
 '1973-Nixon.txt',
 '1977-Carter.txt',
 '1981-Reagan.txt',
 '1985-Reagan.txt',
 '1989-Bush.txt',
 '1993-Clinton.txt',
 '1997-Clinton.txt',
 '2001-Bush.txt',
 '2005-Bush.txt',
 '2009-Obama.txt',
 '2013-Obama.txt',
 '2017-Trump.txt']
```

```
Total number of text file in inaugural corpus: 58
```

- Below are the number of characters in each speech: -

```
Number of charcaters in speach by Mr. Nixon: 9991

Number of charcaters in speach by Mr. Roosevelt: 7571

Number of charcaters in speach by Mr. Kennedy: 7618
```

- Below are the number of words in each speech: -

```
Number of words in speach by Mr. Nixon: 2028

Number of words in speach by Mr. Roosevelt: 1536

Number of words in speach by Mr. Kennedy: 1546
```

- Below are the number of sentences in each speech: -

```
Number of sentences in speach by Mr. Nixon: 69

Number of sentences in speach by Mr. Roosevelt: 68

Number of sentences in speach by Mr. Kennedy: 52
```

2.2) Remove all the stopwords from the three speeches.

Solution: -

Performing basic text pre-processing: -

- Created new DataFrame for the three required speeches as python works better with data in tabular format.

	speech
nixon	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...
roosevelt	On each national day of inauguration since 178...
kennedy	Vice President Johnson, Mr. Speaker, Mr. Chief...

- Converting every character into small case to have uniformity across the text file

	speech
nixon	mr. vice president, mr. speaker, mr. chief jus...
roosevelt	on each national day of inauguration since 178...
kennedy	vice president johnson, mr. speaker, mr. chief...

- Data after removing stopwords:-

	speech
nixon	mr. vice president, mr. speaker, mr. chief jus...
roosevelt	national day inauguration since 1789, people r...
kennedy	vice president johnson, mr. speaker, mr. chief...

We can notice that the stop words like “on”, “each”, “of”….etc are removed from each text files and number of words are reduced.

- Data after removing punctuations: -

	speech
nixon	mr vice president mr speaker mr chief justice ...
roosevelt	national day inauguration since 1789 people re...
kennedy	vice president johnson mr speaker mr chief jus...

We can notice that the punctuations are removed which resulted in reduced number of characters. We have removed punctuations like “,”, “.”….etc.

- Data after stemming the data: -

```
nixon        mr vice presid mr speaker mr chief justic sena...
roosevelt    nation day inaugur sinc 1789 peopl renew sens ...
kennedy      vice presid johnson mr speaker mr chief justic...
```

Stemming the data helps us in reducing inflected (or sometimes derived) words to their word stem, base or root form-generally a written word form. This in-turn helps in reducing the number of characters in text file which makes analysis further easy and smooth.

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)

Solution: -

- Below are the most 3 words used in each text file after pre-processing the data: -

```
The top most three words used in speech by Mr. Nixon: [('us', 26), ('let', 22), ('peace', 19)]

The top most three words used in speech by Mr. Roosevelt: [('nation', 11), ('know', 10), ('spirit', 9)]

The top most three words used in speech by Mr. Kennedy: [('let', 16), ('us', 12), ('world', 8)]
```

From above we can notice that few words like “let”, “know”, “us”, “new” etc was not the part of stopwords and is still the part of text file and appearing as most used words. We have extended our stopwords to remove this form the text files.

- Below are the top 3 used words after extending stopwords and removing them from text files: -

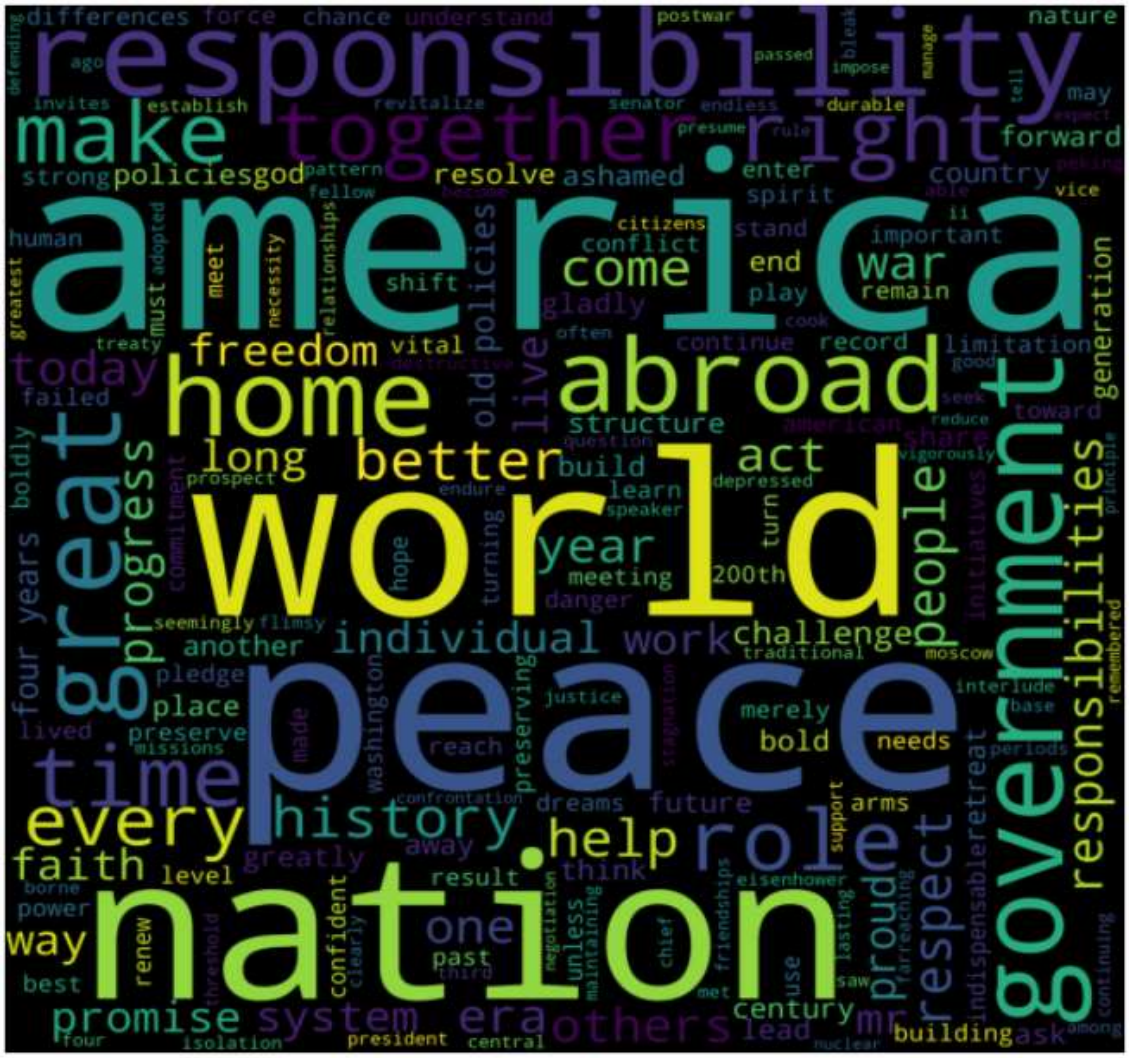
```
The top most three words used in speech by Mr. Nixon: [('peace', 19), ('world', 16), ('america', 13)]
The top most three words used in speech by Mr. Roosevelt: [('nation', 11), ('spirit', 9), ('democracy', 9)]
The top most three words used in speech by Mr. Kennedy: [('world', 8), ('sides', 8), ('pledge', 7)]
```

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

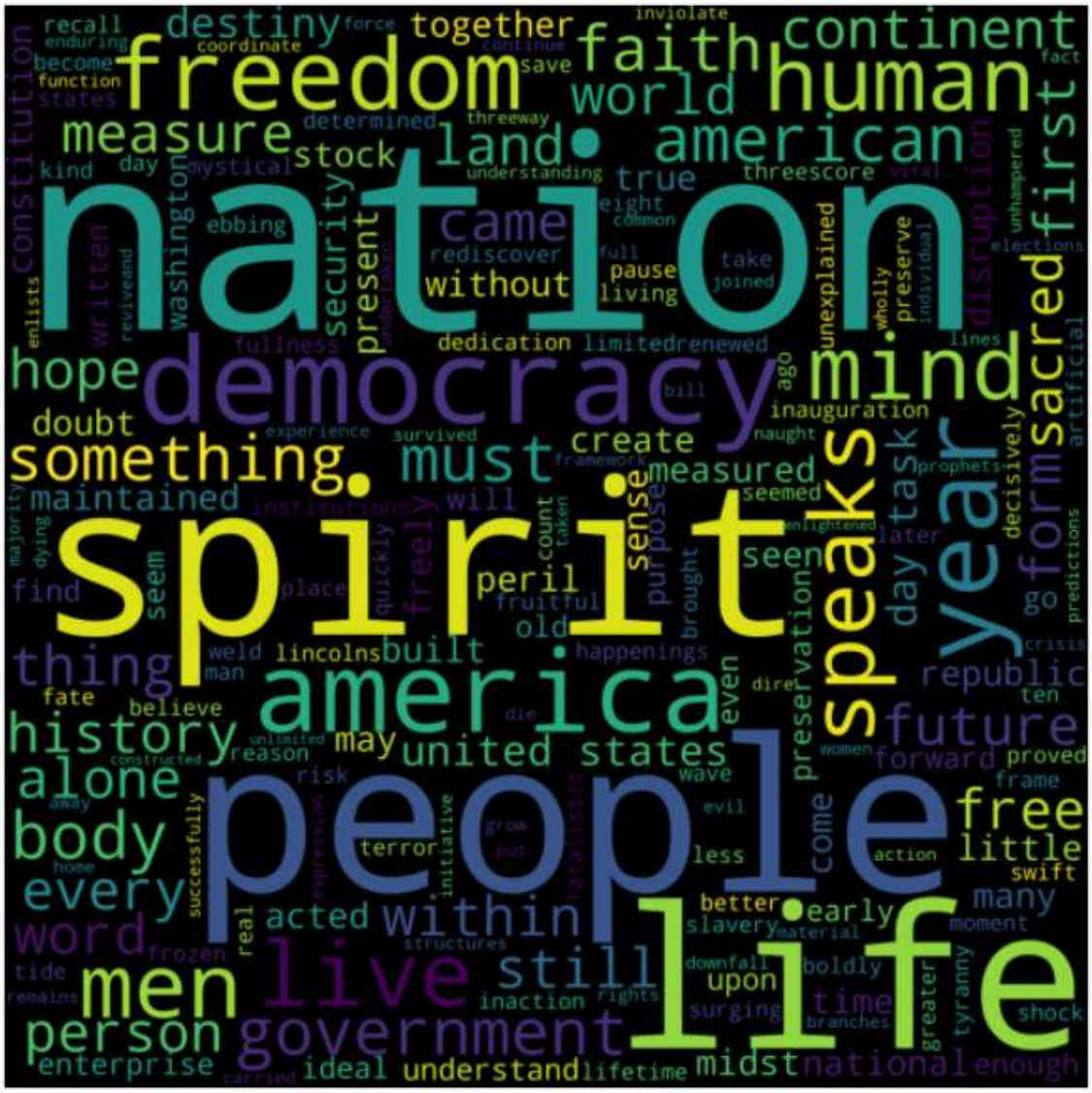
Solution: -

After pre-processing the text files we are able to get the cleaner version of text file and can now create a word cloud for each text file consisting of most words used in each speech and depth of words in the word cloud depends on the frequency its been used: -

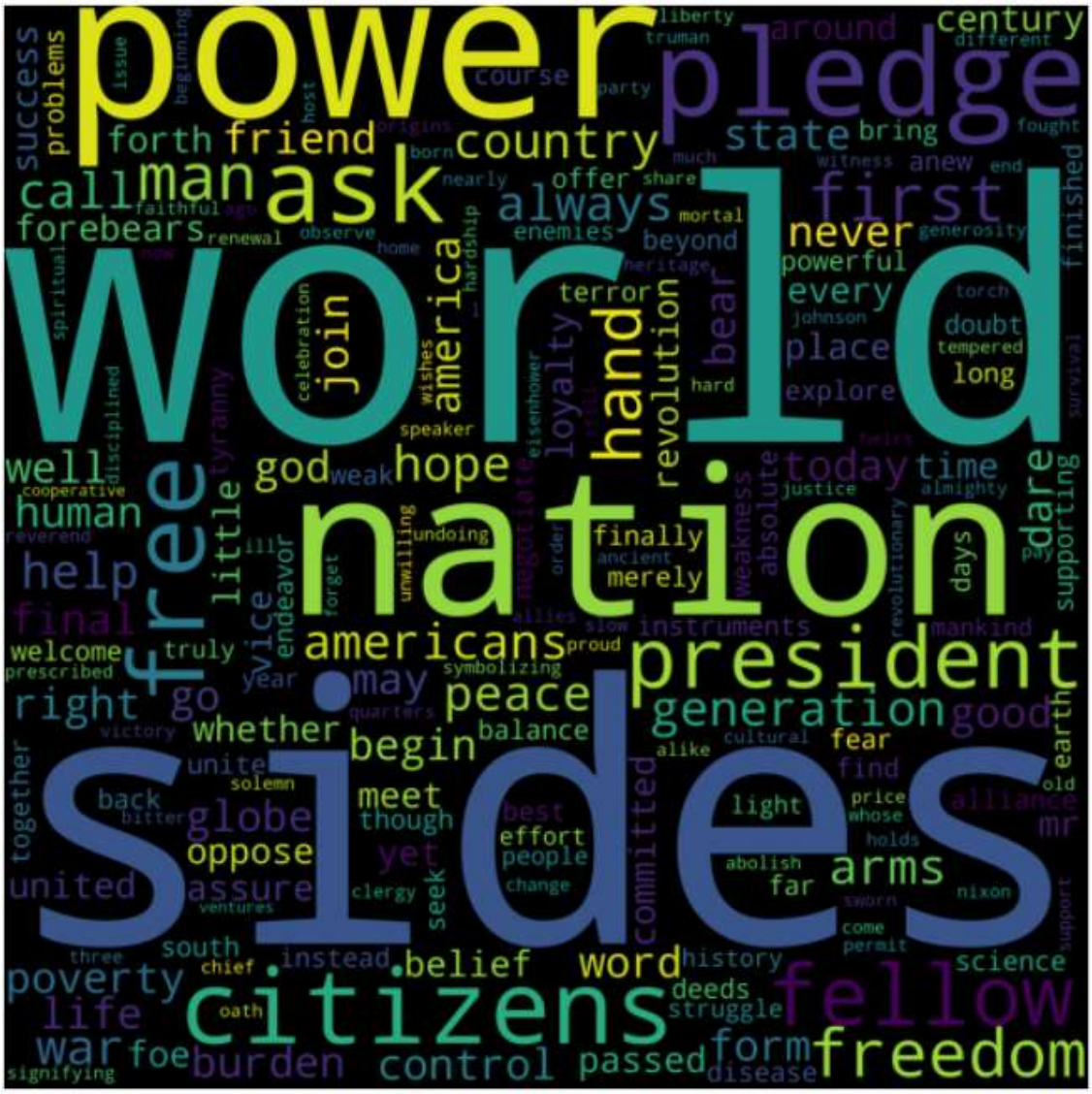
Below is the word-cloud of speech by Mr. Nixon



Below is the word-cloud of speech by Mr. Roosevelt



Below is the word-cloud of speech by Mr. Kennedy



----------END OF PROBLEM—2 ----------