MACHINE LEARNING PROJECT

Data Analysis Report

Prepared By

JAI GOUTHAM

Submitted on
21-02-2021

PROJECT OBJECTIVE

Problem 1: Machine Learning Models

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

1.1 Read the dataset. Do the descriptive statistics and do null value condition check.

1.2 Perform exploratory data analysis. Describe the data briefly. (Data types, shape, EDA). Perform Univariate and Bivariate Analysis. Check for outliers. Interpret the inferences for each.

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)

1.4 Apply Logistic Regression and LDA (Linear Discriminant Analysis). Interpret the inferences of both models

1.5 Apply KNN Model and Naïve Bayes Model. Interpret the inferences of each model

1.6 Model Tuning, Bagging and Boosting.

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 all models on the basis of the performance metrics in a structured tabular manner. Describe on which model is best/optimized

1.8 Based on your analysis and working on the business problem, detail out appropriate insights and recommendations to help the management solve the business objective.

Problem 2:

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

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

2.1 Find the number of characters, words and sentences for the mentioned documents.
2.2 Remove all the stop words from all the three speeches.
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 stop words)
2.4 Plot the word cloud of each of the speeches of the variable. (After removing the stop words).

PROBLEM 1: MACHINE LEARNING

1.1 Read the data and do descriptive statistics and do null value check.

Data Insights and descriptive statistics :

The dataset: “Election.csv” which contains data of 1525 rows and 9 variables namely as follows:

Variable Name	Description
Vote	Party choice: Conservative or Labour
Age	In years
economic.cond.national	Assessment of current national economic conditions, 1 to 5.
economic.cond.household	Assessment of current household economic conditions, 1 to 5.
Blair	Assessment of the Labour leader, 1 to 5.
Hague	Assessment of the Conservative leader, 1 to 5.
Europe	an 11-point scale that measures respondents' attitudes toward European integration. High scores represent ‘Eurosceptic’ sentiment.
political.knowledge	Knowledge of parties' positions on European integration, 0 to 3.
gender	female or male

Description of dataset:

	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

Checking for Missing Values:

The dataset does contain ‘0’ missing/Null values.

```
data_df.isna().sum()
```

```
vote                       0
age                        0
economic_cond_national     0
economic_cond_household    0
Blair                      0
Hague                      0
Europe                     0
political_knowledge        0
gender                     0
dtype: int64
```

Checking for Duplicate Values:

The dataset does contain 8 duplicated entry in the dataset and it is dropped.

```
# Are there any duplicates ?
dups = data_df.duplicated()
print('Number of duplicate rows = %d' % (dups.sum()))

print('Before',data_df.shape)
data_df.drop_duplicates(inplace=True)

dups = data_df.duplicated()
print('Number of duplicate rows = %d' % (dups.sum()))
```

```
Number of duplicate rows = 8
Before (1525, 9)
Number of duplicate rows = 0
```

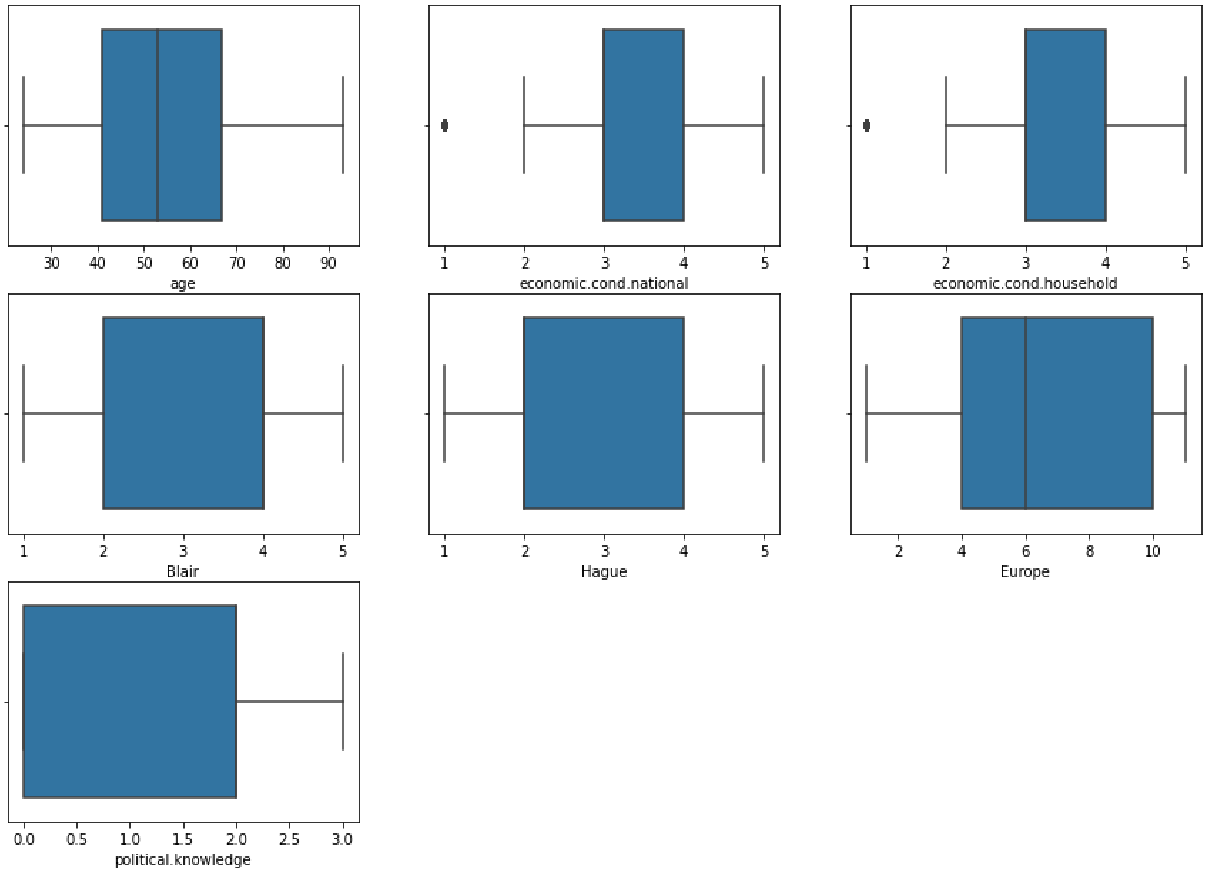
Inference from descriptive statistics:

- The variable ‘Vote’ has two categories labour and conservative
- The average Age of the person is 54, with minimum age of 24 and maximum age of 93.
- There are no missing or null values present in the dataset.
- There are 8 duplicate entries present in the dataset and it is removed from the dataset.

1.2 Perform exploratory data analysis. Describe the data briefly. (Data types, shape, EDA). Perform Univariate and Bivariate Analysis. Check for outliers. Interpret the inferences for each.

Boxplot:

Using the Boxplot in a dataset we can able to find the outliers, spread of values, median, range, etc., (outliers are the extreme values present in the dataset)

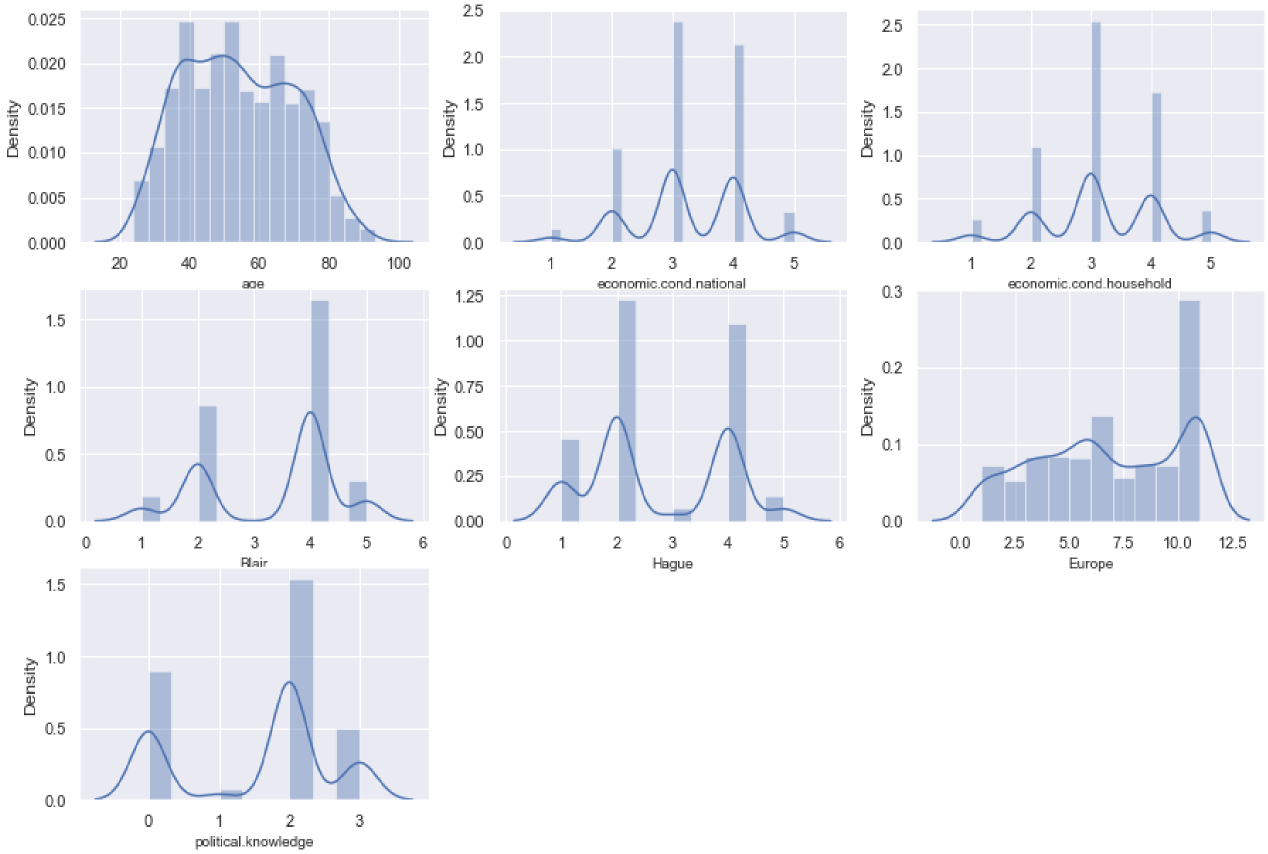


Inference from boxplot and Outlier Treatment:

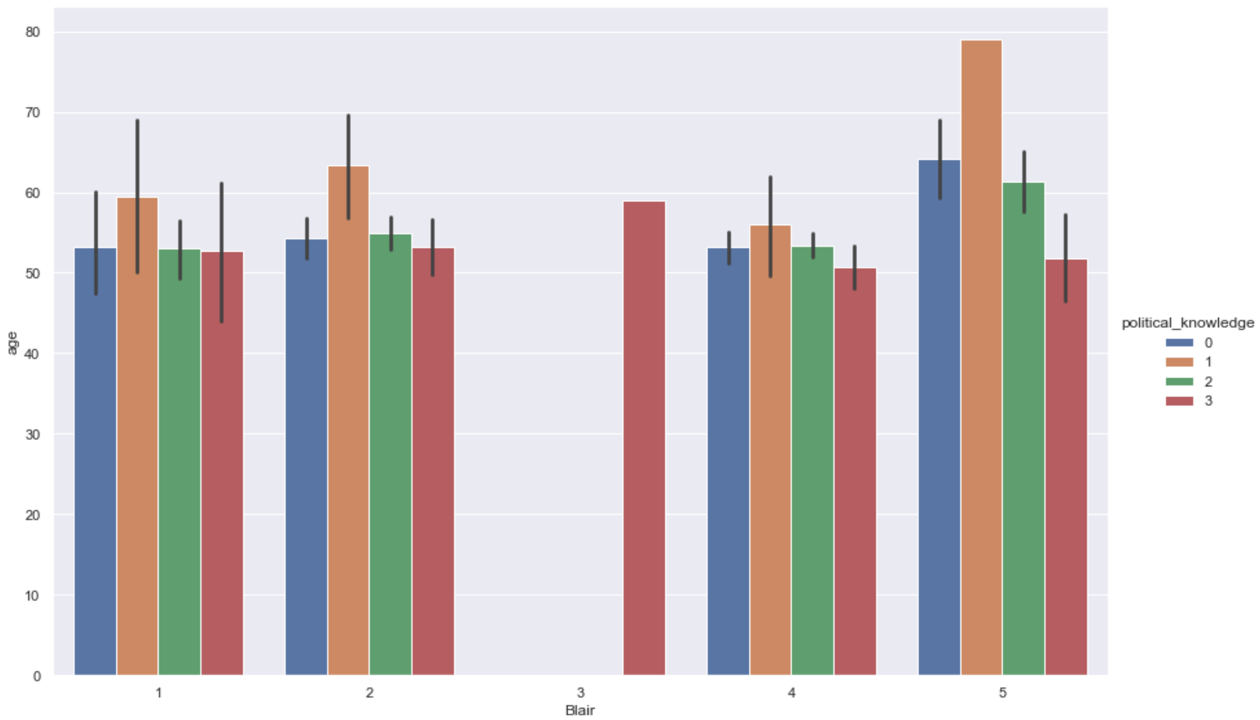
From the above Boxplots for all the variables, we can conclude that most of the variables have no outliers. There are only two outliers present in the variables like economic_cond_national and economic_cond_household, so significantly it may not impact on our dataset, so no need to go for outlier treatment.

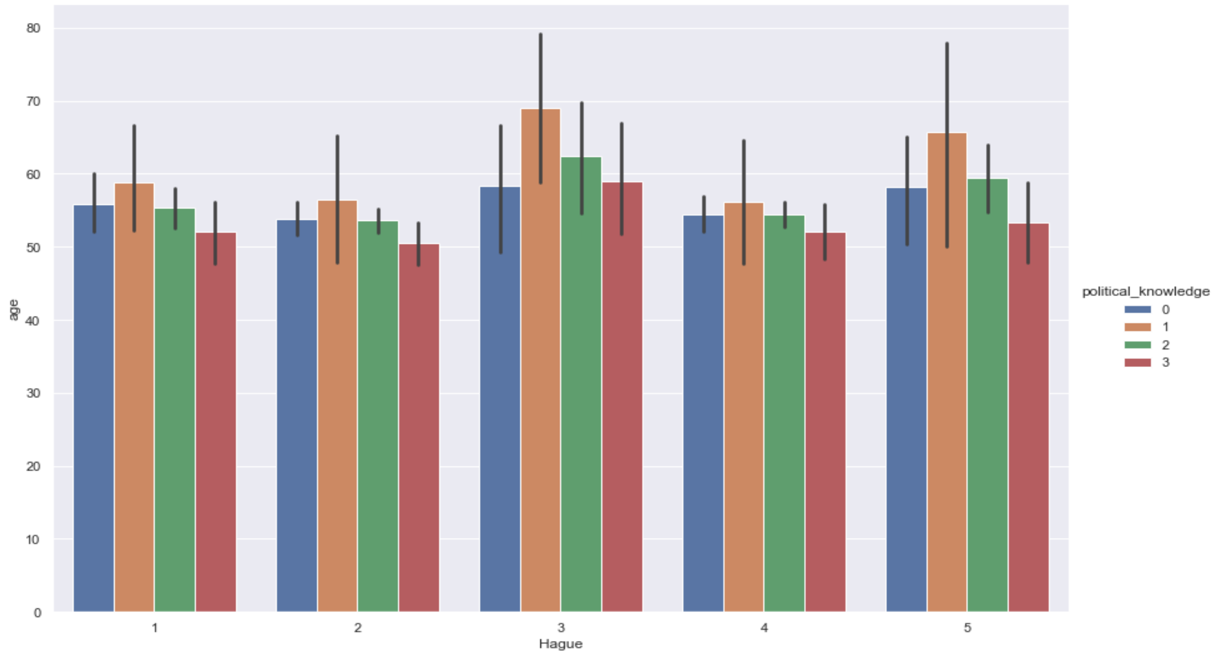
- More than 50% of people with age range have approximately (40-70) years.
- Most of the national economic condition is in the range of 3-4 points.
- The national economic household condition also in the range of 3-4 points.
- The European integration sentiment scale which has approx. 65% in higher side.
- Both the political leader Blair and Hague has an assessment range of 2-4 points.

Histogram:



Barplot:



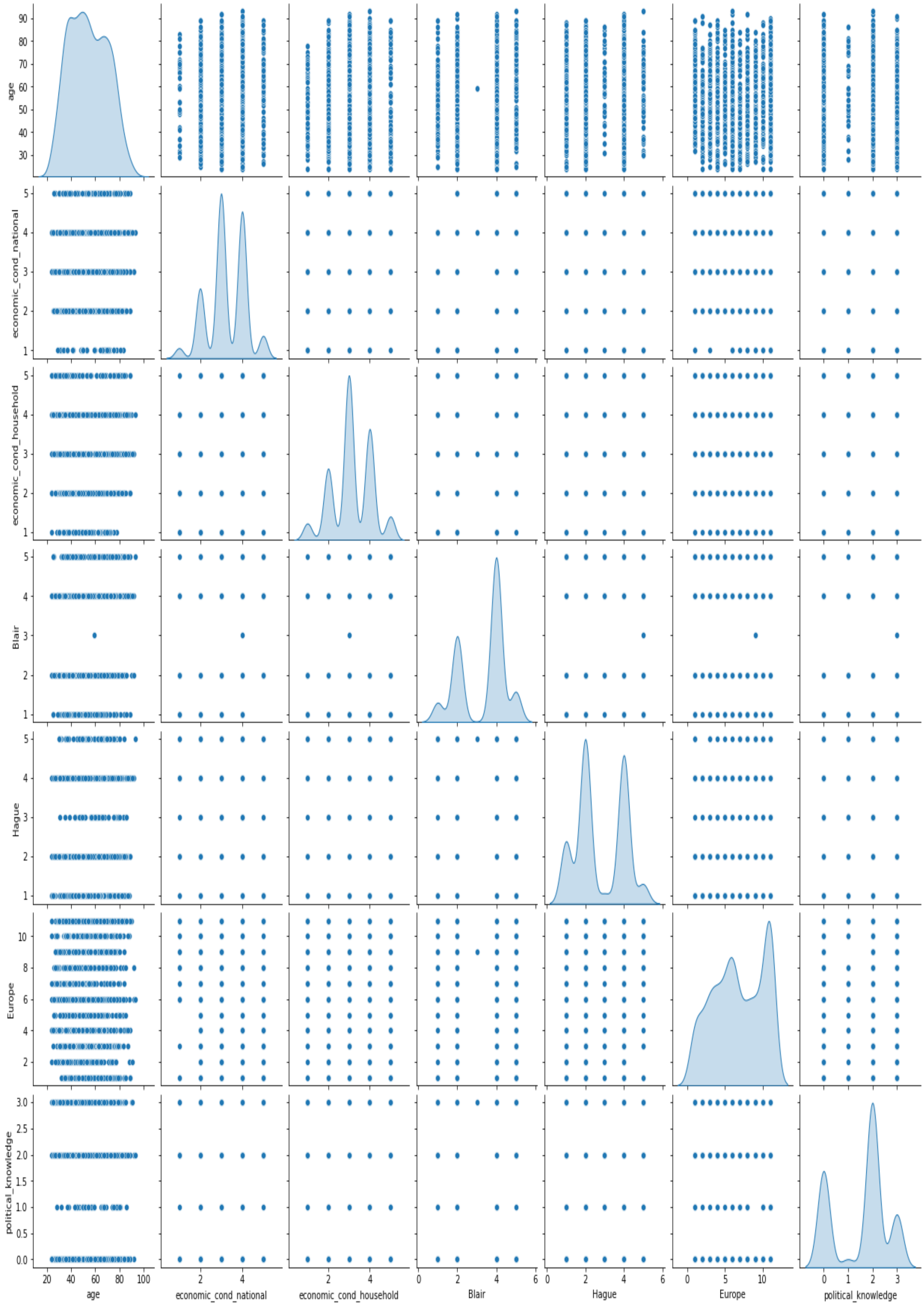


Inference from Univariate analysis:

From the above all plots for all the variables, we can conclude the below points:

- More than 70% of people with age range have approximately (35-80) years.
- The Age group of 45-50 has highest participants of around 230 peoples.
- (420 nos) 27.54% of participants have ultimate scepticism towards European integration.
- More than 1250 participants has rated higher scale on national economic condition has range of 3-4 points.
- More than 1200 participants has rated higher scale on national household condition has range of 3-4 points.
- (Around 850 nos.) 55.74% of participants has given above average Assessment of 3.7 to 4.3 for Labour leader
- Interestingly (Around 1000 no's) 65.57% of participants have given high assessment of 3.7-5.0 for labour leader.
- (Around 625 nos.) 40.98% of participants have only given above average rating of 3-5.
- Around (1025 nos) 67.21% of participants are aware of the parties position on European integration.
- Around (950 nos) 62.30% of participants have given rating of more than 6 in the scale, which shows that majority of the participants are much sceptical about European integration.

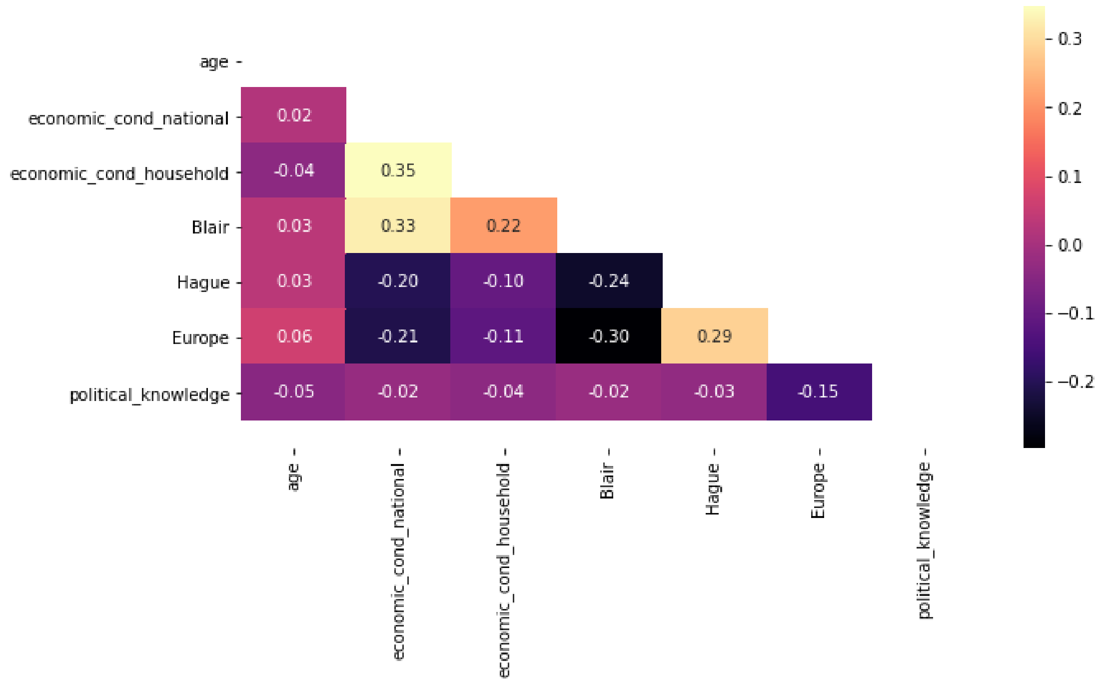
Pairplot:



age
economic_cond_national
economic_cond_household
Blair
Hague
Europe
political_knowledge

Heat Map

The Heat Map shows the relationship between different variables in our dataset. This graph can help us to check for any correlations between different variables.



Inference from Multivariate analysis:

- **There is no strong multicollinearity among variables**
- Ratings of household economic condition national economic condition have some meaningful positive correlation (0.35)
- Participants giving high rating to conservative party are to certain extent eurosceptical. (positive correlation of 0.29)
- Participants giving high rating to national economic condition are supporters of labour party.(positive correlation of 0.33)
- None of variables are highly correlated with each other
- A rating of 0, 2 & 3 on Knowledge of parties' positions on European integration has not been influenced by different age groups.
- The Eurosceptic sentiments have spread across the complete spectrum of age groups.
- Participants Eurosceptic sentiment has not influenced their assessments on national and household economic conditions

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)

Before Encoding:

```
data_df.head()
```

	vote	age	economic_cond_national	economic_cond_household	Blair	Hague	Europe	political_knowledge	gender
0	Labour	43	3	3	4	1	2	2	female
1	Labour	36	4	4	4	4	5	2	male
2	Labour	35	4	4	5	2	3	2	male
3	Labour	24	4	2	2	1	4	0	female
4	Labour	41	2	2	1	1	6	2	male

After Encoding:

```
data_df=pd.get_dummies(data_df,columns=['gender'],drop_first=True)
data_df.head()
```

	vote	age	economic_cond_national	economic_cond_household	Blair	Hague	Europe	political_knowledge	gender_male
0	Labour	43	3	3	4	1	2	2	0
1	Labour	36	4	4	4	4	5	2	1
2	Labour	35	4	4	5	2	3	2	1
3	Labour	24	4	2	2	1	4	0	0
4	Labour	41	2	2	1	1	6	2	1

```
for col in data_df.columns:
    if data_df[col].dtype=='object':
        data_df[col]=pd.Categorical(data_df[col]).codes
```

```
data_df.tail()
```

	vote	age	economic_cond_national	economic_cond_household	Blair	Hague	Europe	political_knowledge	gender_male
1520	0	67	5	3	2	4	11	3	1
1521	0	73	2	2	4	4	8	2	1
1522	1	37	3	3	5	4	2	2	1
1523	0	61	3	3	1	4	11	2	1
1524	0	74	2	3	2	4	11	0	0

Splitting the dataset and dropping the dependent variable (vote) for X.

```
# Seperating INDEPENDENT AND DEPENDENT VARIABLE

X=data_df.drop('vote',axis=1)
y=data_df.pop('vote')

y.value_counts()
```

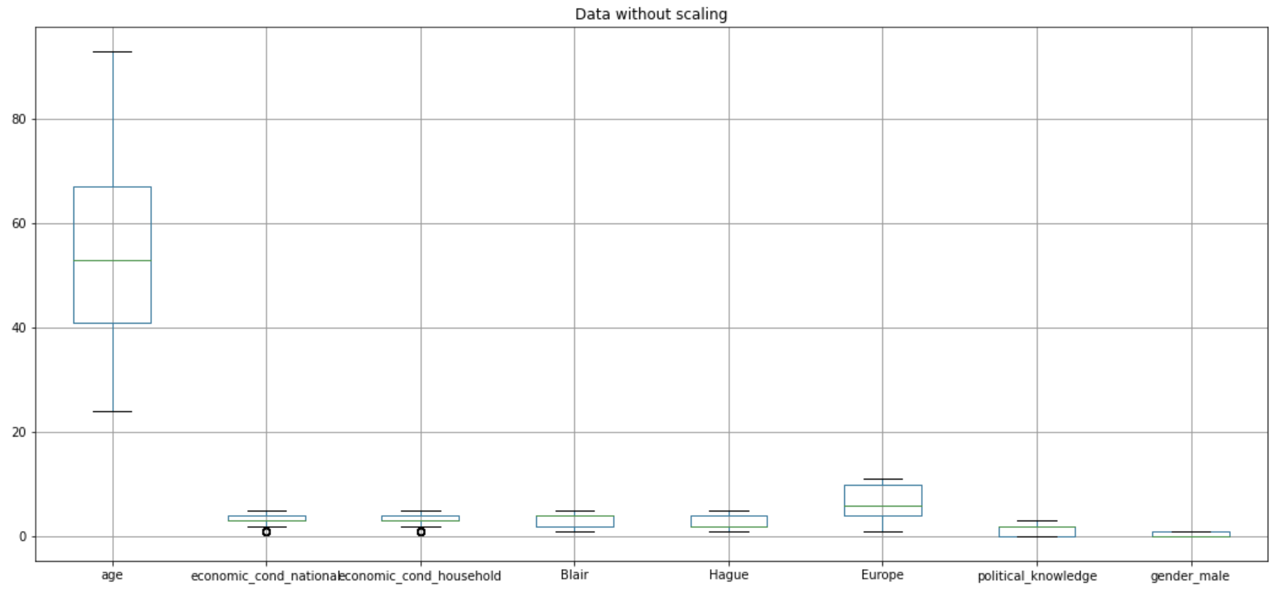
```
1    1057
0     460
Name: vote, dtype: int64
```

Scaling:

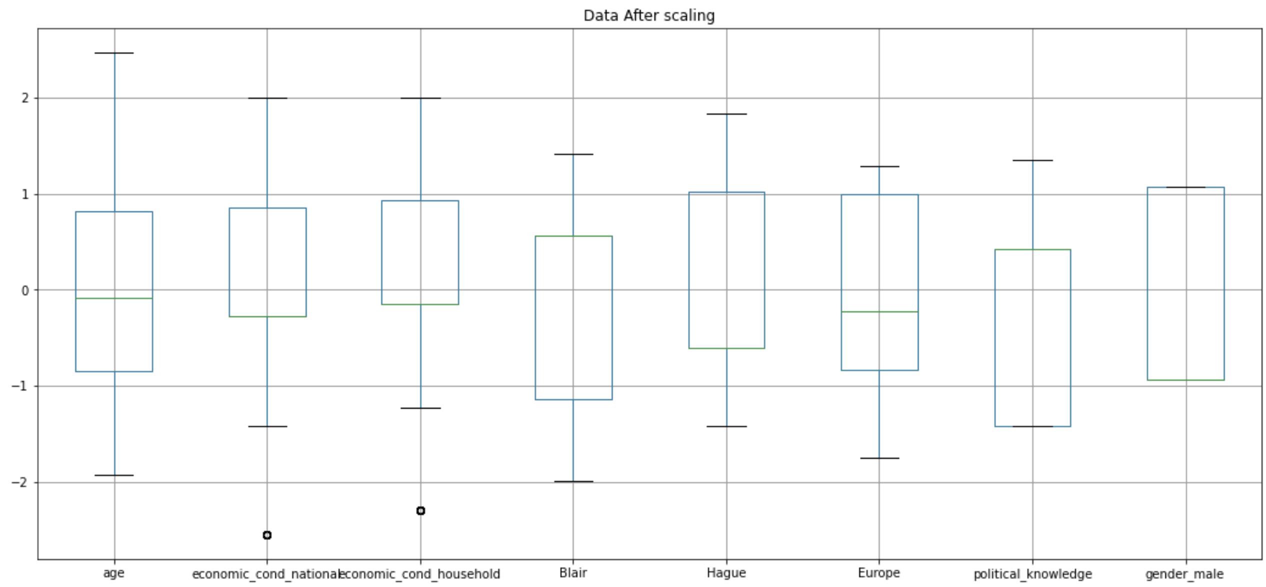
Scaling is required for the models like Logistic regression, SVM, ANN etc., since it works on gradient descent based algorithm. And also for KNN, SVM it is necessary to scale the data, as it a distance-based algorithm (typically based on Euclidean distance). Scaling the data gives similar weightage to all the variables.

For Tree based models like Decision Tree, scaling is not required since it works on splitting a node based on a single feature. The decision tree splits a node on a feature that increases the homogeneity of the node. This split on a feature is not influenced by other features.

So, we are applying Standardization on the encoded variables by using the scipy.stats library and using Z-score (Standardization).



After Scaling:



```
Xs=X.copy()
from scipy.stats import zscore
Xs=Xs.apply(zscore)
Xs.head()
```

	age	economic_cond_national	economic_cond_household	Blair	Hague	Europe	political_knowledge	gender_male
0	-0.716161	-0.278185	-0.148020	0.565802	-1.419969	-1.437338	0.423832	-0.936736
1	-1.162118	0.856242	0.926367	0.565802	1.014951	-0.527684	0.423832	1.067536
2	-1.225827	0.856242	0.926367	1.417312	-0.608329	-1.134120	0.423832	1.067536
3	-1.926617	0.856242	-1.222408	-1.137217	-1.419969	-0.830902	-1.421084	-0.936736
4	-0.843577	-1.412613	-1.222408	-1.988727	-1.419969	-0.224465	0.423832	1.067536

Splitiing the data into Training and testing set

```
X_train,X_test,y_train,y_test=train_test_split(Xs,y,test_size=0.30,random_state=1)
```

1.4 Apply Logistic Regression and LDA (Linear Discriminant Analysis). Interpret the inferences of both models

1.4.1 Logistic regression (Base model)

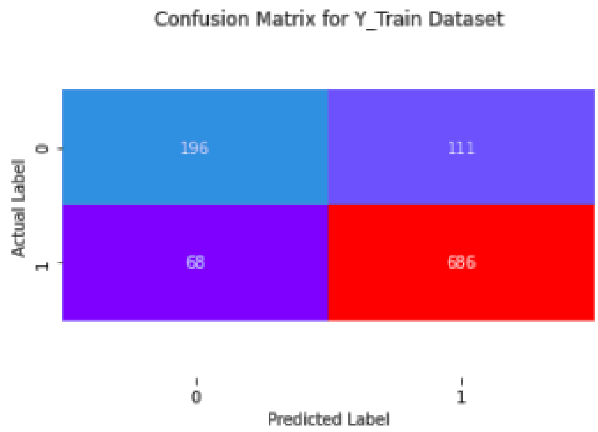
```
from sklearn.linear_model import LogisticRegression

lr=LogisticRegression(penalty='l2',tol=0.0001,random_state=1,solver='lbfgs',max_iter=100)

lr.fit(X_train,y_train)
```

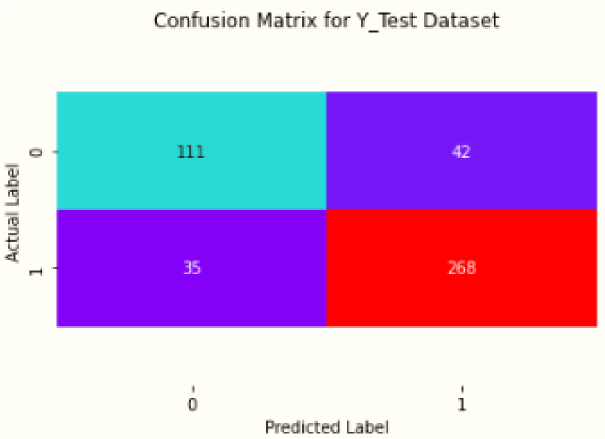
```
LogisticRegression(random_state=1)
```

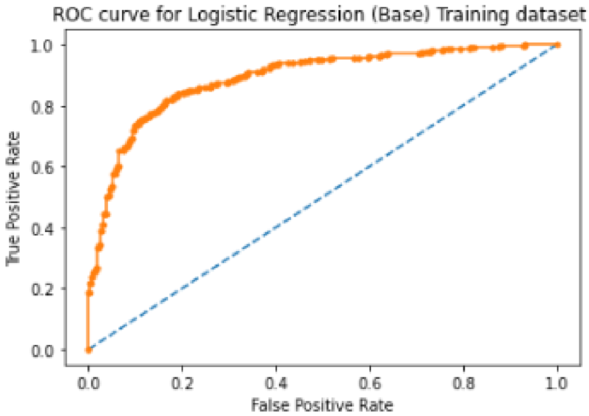
Predicting on Training and Test dataset

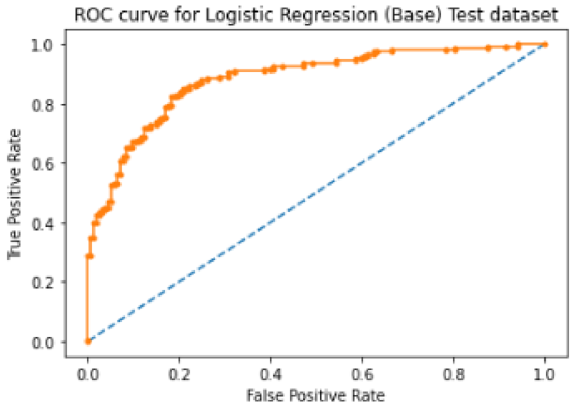
```
ytrain_predict = lr.predict(X_train)
ytest_predict = lr.predict(X_test)
```

```
The accuracy score for the Training dataset is: 0.8312912346842601


The classification report for the Training dataset is:
              precision    recall  f1-score   support

           0       0.74      0.64      0.69       307
           1       0.86      0.91      0.88       754

    accuracy                           0.83      1061
   macro avg       0.80      0.77      0.79      1061
weighted avg       0.83      0.83      0.83      1061



The accuracy score for the Testing dataset is: 0.831140350877193


The classification report for the Testing dataset is:
              precision    recall  f1-score   support

           0       0.76      0.73      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
```







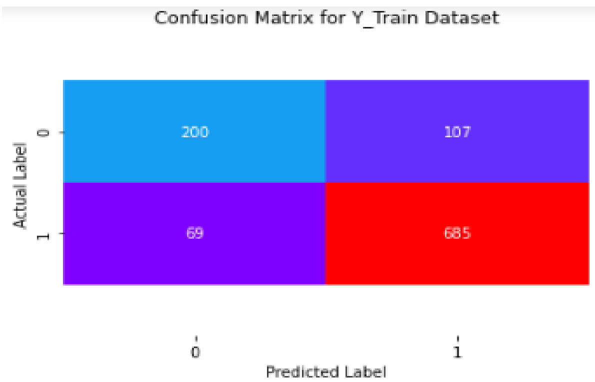


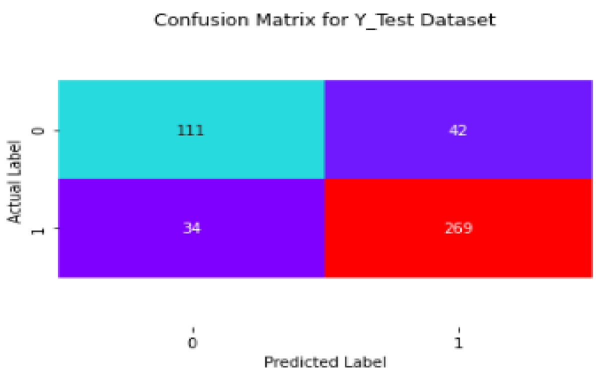
1.4.2 LDA Linear Discriminant Analysis (Base Model)

```
#Build LDA Model
lda = LinearDiscriminantAnalysis()
lda=lda.fit(X_train,y_train)
```

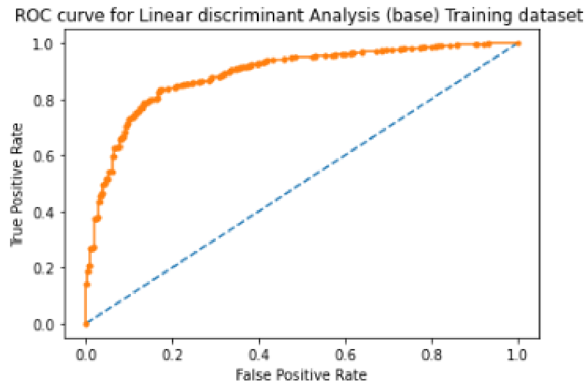
```
# Training Data Class Prediction with a cut-off value of 0.5
ytrain_predict = lda.predict(X_train)

# Test Data Class Prediction with a cut-off value of 0.5
ytest_predict = lda.predict(X_test)
```

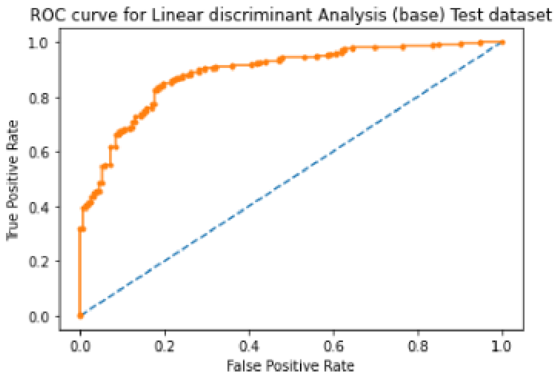
```
# Training Data Probability Prediction
ytrain_pred_prob = lda.predict_proba(X_train)

# Test Data Probability Prediction
ytest_pred_prob = lda.predict_proba(X_test)
```





```
The accuracy score for the Training dataset is: 0.8341187558906692


The classification report for the Training dataset is:
              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



The accuracy score for the Testing dataset is: 0.8333333333333334


The classification report for the Testing dataset is:
              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
```





Inference from both model (Logit and LDA):

Both the models have performed well in both the Training and Test data. While the model results between training and test sets are similar, indicating no under or over fitting issues.

Class 1 represents Labour party which contains 70% of voting data and Class 0 represents Conservative party which contains 30% voting data.

- Both the model score of the Training data is 83% and Test data is 83%
- AUC score - Training and Test data while for Logistic regression is 89% and 88%
- AUC score - Training and Test data while for Linear discriminate analysis is 88% and 88%.

The below table which shows the Class 1 (Labour party) the model LDA which performs well when compare to Logistic regression.

Model	Accuracy		Precision		Recall		F1 Score		AUC Score	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Logit	0.83	0.83	0.86	0.86	0.91	0.88	0.88	0.87	0.890	0.883
LDA	**0.83**	**0.83**	**0.86**	**0.86**	**0.91**	**0.89**	**0.89**	**0.88**	**0.889**	**0.888**

The below table which shows the Class 0 (Conservative party) the model LDA which performs well when compare to Logistic regression.

Model	Accuracy		Precision		Recall		F1 Score		AUC Score	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Logit	0.83	0.83	0.74	0.76	0.64	0.73	0.69	0.74	0.890	0.883
LDA	**0.83**	**0.83**	**0.74**	**0.77**	**0.65**	**0.73**	**0.69**	**0.74**	**0.889**	**0.888**

On comparing both the model the **Linear Discriminant Analysis** model perform well on testing data with accuracy 83%, precision 77%, Recall 73%.

1.5 Apply KNN Model and Naïve Bayes Model. Interpret the inferences of each model

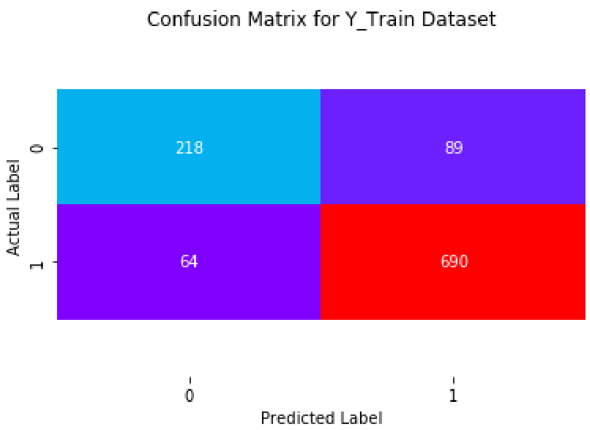
1.5.1.1 KNN (Base Model)

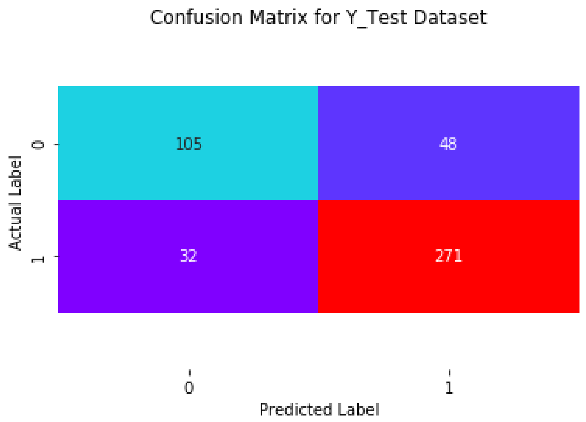
```
from sklearn.neighbors import KNeighborsClassifier

knn=KNeighborsClassifier(n_neighbors=5,weights='uniform',algorithm='auto',leaf_size=30,
                         p=2,metric='minkowski',metric_params=None,n_jobs=None,)
knn.fit(X_train,y_train)
```

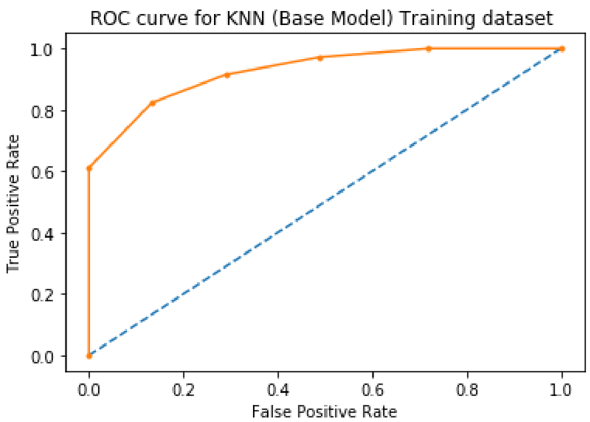
```
KNeighborsClassifier()
```

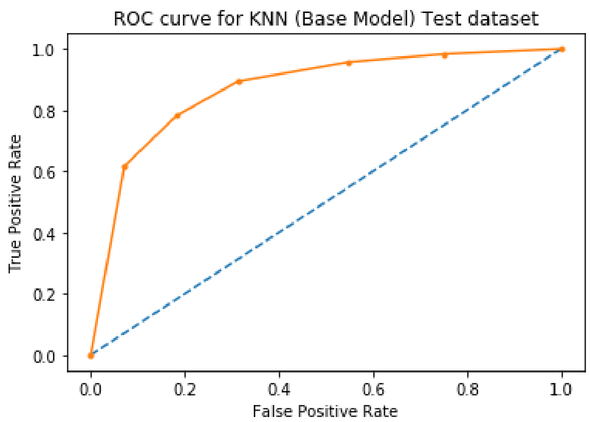
```
ytrain_predict = knn.predict(X_train)
ytest_predict = knn.predict(X_test)
```

```
The accuracy score for the Training dataset is: 0.8557964184731386


The classification report for the Training dataset is:
              precision    recall  f1-score   support

           0       0.77      0.71      0.74       307
           1       0.89      0.92      0.90       754

    accuracy                           0.86      1061
   macro avg       0.83      0.81      0.82      1061
weighted avg       0.85      0.86      0.85      1061



The accuracy score for the Testing dataset is: 0.8245614035087719


The classification report for the Testing dataset is:
              precision    recall  f1-score   support

           0       0.77      0.69      0.72       153
           1       0.85      0.89      0.87       303

    accuracy                           0.82       456
   macro avg       0.81      0.79      0.80       456
weighted avg       0.82      0.82      0.82       456
```









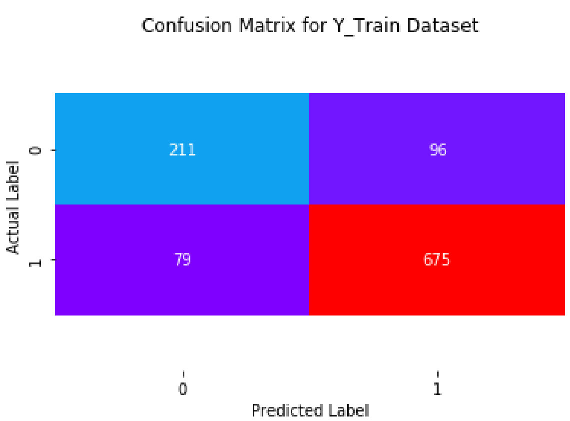
1.5.2.1 Naive Bayes (Base Model)

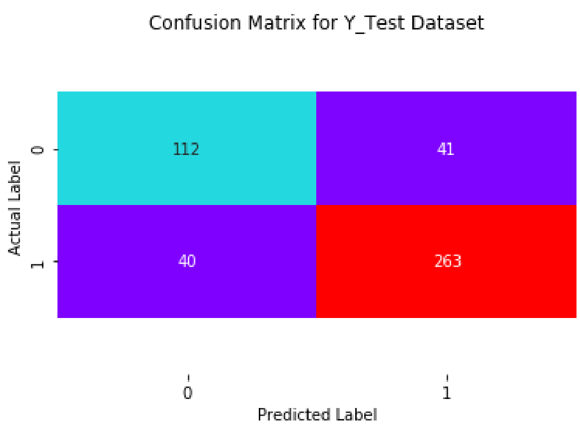
```python
from sklearn.naive_bayes import GaussianNB

nb=GaussianNB()

nb.fit(X_train,y_train)
```

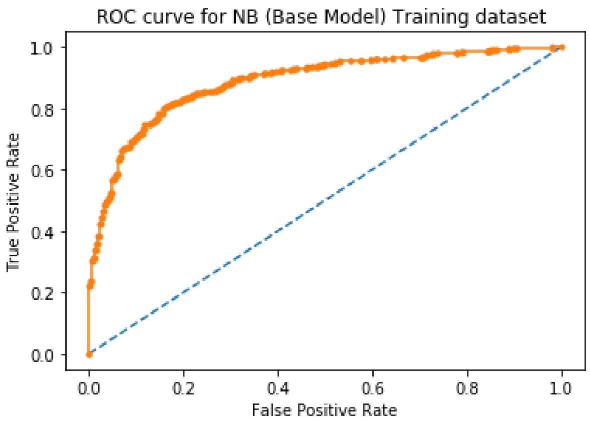
```
GaussianNB()
```

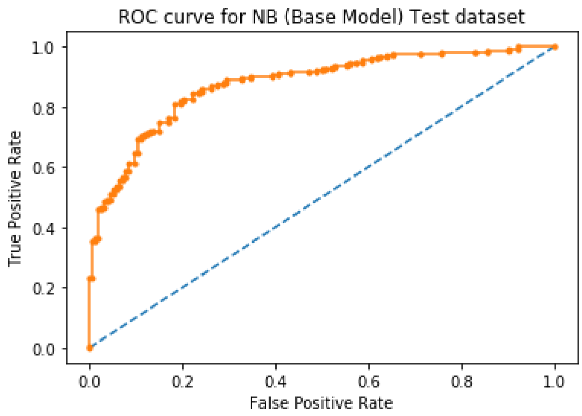
```python
ytrain_predict=nb.predict(X_train)
ytest_predict=nb.predict(X_test)
```

```
The accuracy score for the Training dataset is: 0.8350612629594723


The classification report for the Training dataset is:
              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



The accuracy score for the Testing dataset is: 0.8223684210526315


The classification report for the Testing dataset is:
              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
```









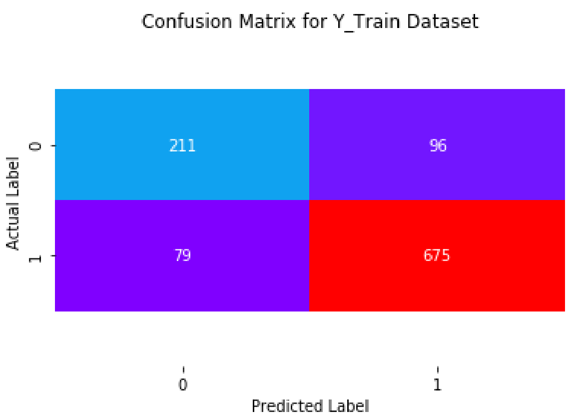
1.5.3.1 Support Vector Machine (Base Model)

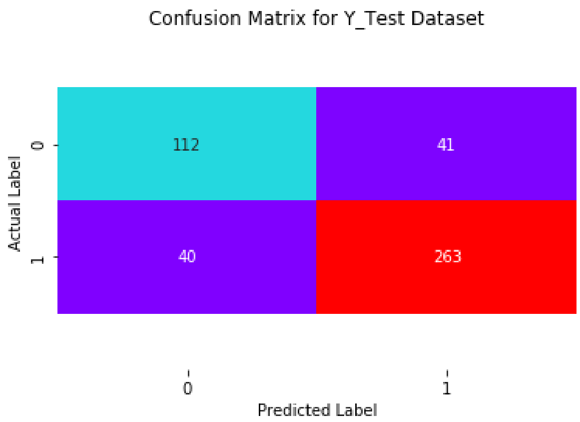
```
from sklearn import svm

sv=svm.SVC(C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0,
           shrinking=True, tol=0.001, cache_size=200,  class_weight=None,  max_iter=-1,
            decision_function_shape='ovr', random_state=1,probability=True)

sv.fit(X_train,y_train)
```

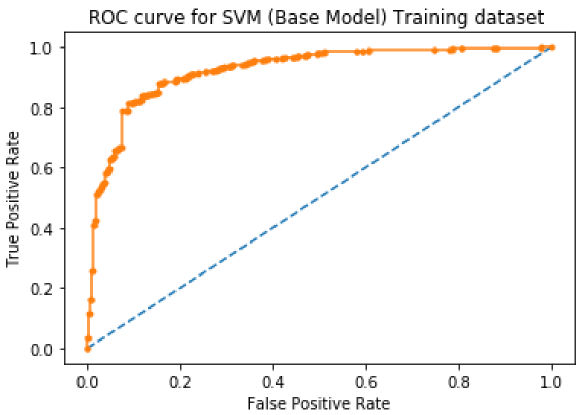
```
SVC(probability=True, random_state=1)
```

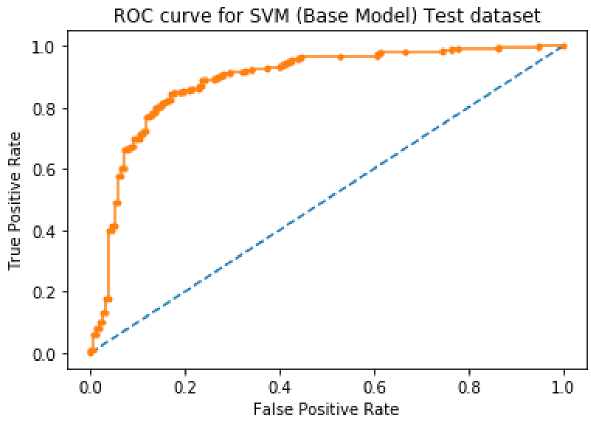
```
ytrain_predict=nb.predict(X_train)
ytest_predict=nb.predict(X_test)
```

```
The accuracy score for the Training dataset is: 0.8671065032987747


The classification report for the Training dataset is:
              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



The accuracy score for the Testing dataset is: 0.8399122807017544


The classification report for the Testing dataset is:
              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
```









Inference from both model (KNN, Naïve Bayes, SVM):

All the models have performed well in both the Training and Test data. While the model results between training and test sets are similar, indicating no under or over fitting issues.

Class 1 represents Labour party which contains 70% of voting data and Class 0 represents Conservative party which contains 30% voting data.

- AUC score - Training and Test data while for KNN is 92% and 87%
- AUC score - Training and Test data while for NB is 89% and 87%
- AUC score - Training and Test data while for SVM is 92% and 88%.

The output results for Labour Party as follows

Model	Accuracy		Precision		Recall		F1 Score		AUC Score	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
KNN	**0.86**	**0.82**	**0.89**	**0.85**	**0.92**	**0.89**	**0.90**	**0.87**	**0.927**	**0.870**
NB	0.84	0.82	0.88	0.87	0.90	0.87	0.89	0.87	0.888	0.876
SVM	0.84	0.82	0.88	0.87	0.90	0.87	0.89	0.87	0.923	0.888

The output results for Conservative Party as follows

Model	Accuracy		Precision		Recall		F1 Score		AUC Score	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
KNN	**0.86**	**0.82**	**0.77**	**0.77**	**0.71**	**0.69**	**0.74**	**0.72**	**0.927**	**0.870**
NB	0.84	0.82	0.73	0.74	0.69	0.73	0.71	0.73	0.888	0.876
SVM	**0.84**	**0.83**	**0.73**	**0.74**	**0.69**	**0.73**	**0.71**	**0.73**	**0.923**	**0.888**

On comparing both the model the **KNN and SVM** model perform well on testing data with accuracy 83%, precision 77%, Recall 73%.

1.6 Model Tuning, Bagging and Boosting.

1.4.1.1 Logistic Regression (Model Tuning)

```
grid={'penalty':['l2','l1','none'],
      'solver':['saga','lbfgs','newton-cg'],
      'tol':[0.0001,0.00001],
      'C':[0.01,0.1,1.0,1.5,2,5,10,25],
      'max_iter':[100,1000,10000]}

lr_model = LogisticRegression(max_iter=10000,n_jobs=2,random_state=1)

grid_search = GridSearchCV(estimator = lr_model, param_grid = grid, cv =10 ,n_jobs=-1,scoring='f1')
```

```
grid_search.fit(X_train, y_train)
```

```
GridSearchCV(cv=10,
             estimator=LogisticRegression(max_iter=10000, n_jobs=2,
                                          random_state=1),
             n_jobs=-1,
             param_grid={'C': [0.01, 0.1, 1.0, 1.5, 2, 5, 10, 25],
                         'max_iter': [100, 1000, 10000],
                         'penalty': ['l2', 'l1', 'none'],
                         'solver': ['saga', 'lbfgs', 'newton-cg'],
                         'tol': [0.0001, 1e-05]},
             scoring='f1')
```

```
print(grid_search.best_params_,'\n')
print(grid_search.best_estimator_)
```

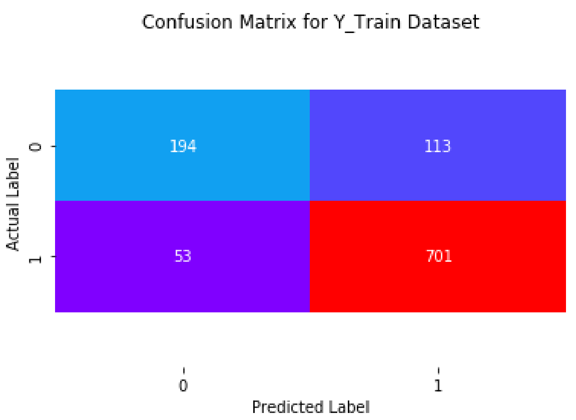
```
{'C': 0.1, 'max_iter': 100, 'penalty': 'l1', 'solver': 'saga', 'tol': 0.0001}

LogisticRegression(C=0.1, n_jobs=2, penalty='l1', random_state=1, solver='saga')
```

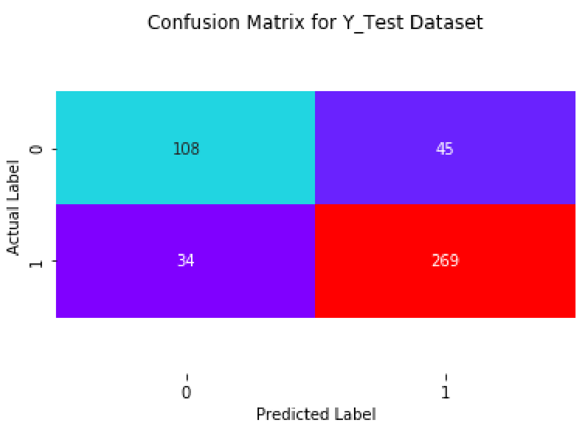
```
best_model = grid_search.best_estimator_
```

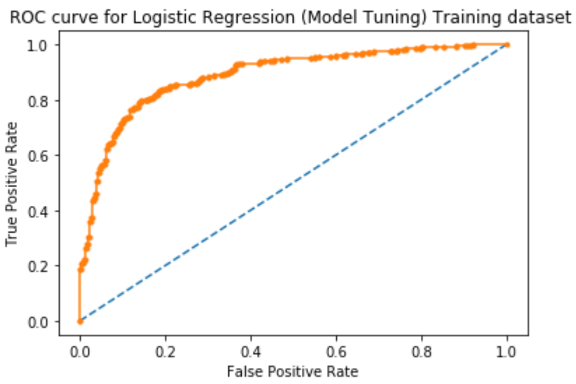
Predicting on Training and Test dataset

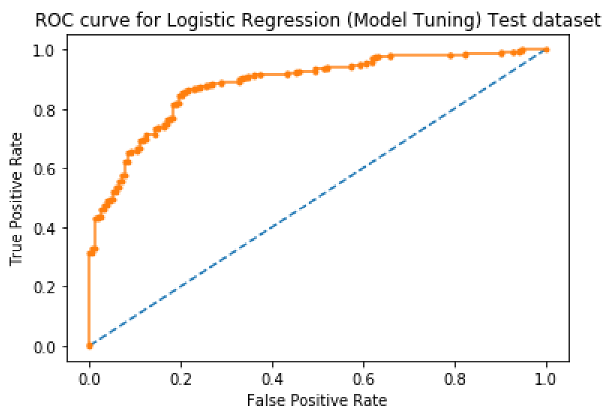
```
ytrain_predict = best_model.predict(X_train)
ytest_predict = best_model.predict(X_test)
```

```
The accuracy score for the Training dataset is: 0.8435438265786993


The classification report for the Training dataset is:
              precision    recall  f1-score   support

           0       0.79      0.63      0.70       307
           1       0.86      0.93      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



The accuracy score for the Testing dataset is: 0.8267543859649122


The classification report for the Testing dataset is:
              precision    recall  f1-score   support

           0       0.76      0.71      0.73       153
           1       0.86      0.89      0.87       303

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









1.4.2 LDA Linear Discriminant Analysis (Model Tuning)

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

lda_estimator=LinearDiscriminantAnalysis(shrinkage='auto')

lda_param={'solver':['svd','lsqr','eigen'],
    'n_components':[1,3,5],
    'tol':[0.00001,0.00001,0.0001,0.001,0.01]}

clf=GridSearchCV(estimator=lda_estimator,
    param_grid=lda_param,
    n_jobs=-1,
    cv=10,)

```

```
lda_model=clf.fit(X_train,y_train)
lda_model
```

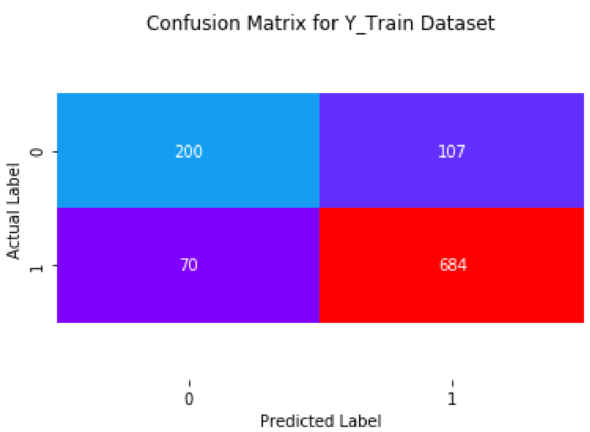
```
GridSearchCV(cv=10, estimator=LinearDiscriminantAnalysis(shrinkage='auto'),
             n_jobs=-1,
             param_grid={'n_components': [1, 3, 5],
                         'solver': ['svd', 'lsqr', 'eigen'],
                         'tol': [1e-05, 1e-05, 0.0001, 0.001, 0.01]})
```

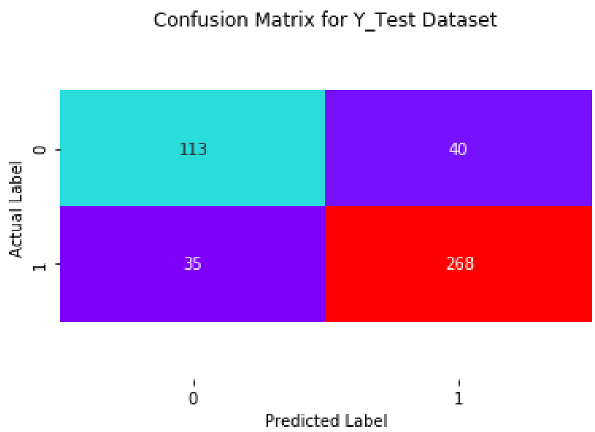
```
clf.best_params_
```

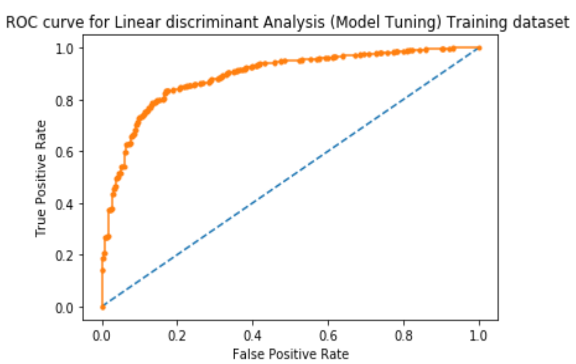
```
{'n_components': 1, 'solver': 'lsqr', 'tol': 1e-05}
```

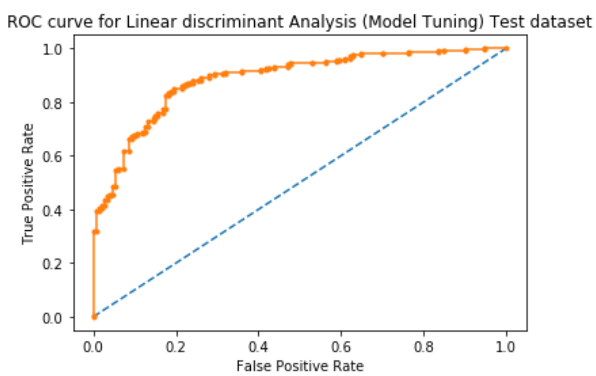
```
lda_grid=clf.best_estimator_
lda_grid
```

```
LinearDiscriminantAnalysis(n_components=1, shrinkage='auto', solver='lsqr',
                           tol=1e-05)
```









```
The accuracy score for the Training dataset is: 0.8331762488218661


The classification report for the Training dataset is:
              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



The accuracy score for the Testing dataset is: 0.8355263157894737


The classification report for the Testing dataset is:
              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
```

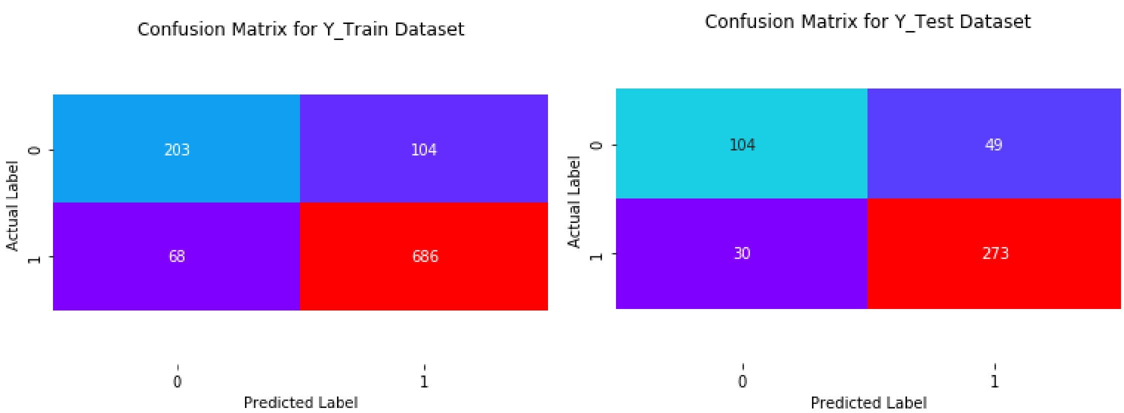
1.5.1.2 KNN (Model Tuning) ¶

```
clf=KNeighborsClassifier()

knn_param={'n_neighbors':[20,25,30,40],
     'leaf_size':[20,30,40],
     'p':[1,2],}

knn_model=GridSearchCV(clf,param_grid=knn_param,cv=10)
knn_model.fit(X_train,y_train)
```

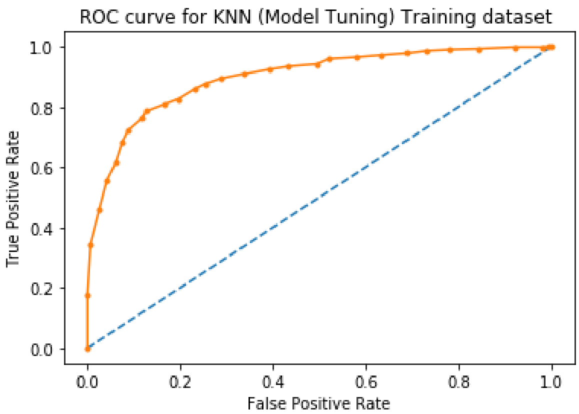
```
GridSearchCV(cv=10, estimator=KNeighborsClassifier(),
             param_grid={'leaf_size': [20, 30, 40],
                         'n_neighbors': [20, 25, 30, 40], 'p': [1, 2]})
```

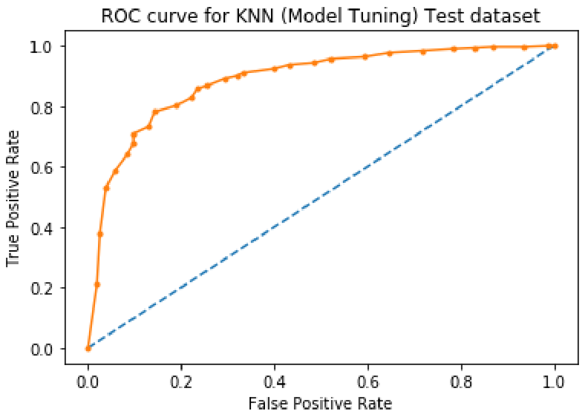
```
knn_model.best_params_
```

```
{'leaf_size': 20, 'n_neighbors': 30, 'p': 1}
```



```
The accuracy score for the Training dataset is: 0.8378887841658812


The classification report for the Training dataset is:
              precision    recall  f1-score   support

           0       0.75      0.66      0.70       307
           1       0.87      0.91      0.89       754

    accuracy                           0.84      1061
   macro avg       0.81      0.79      0.80      1061
weighted avg       0.83      0.84      0.83      1061



The accuracy score for the Testing dataset is: 0.8267543859649122


The classification report for the Testing dataset is:
              precision    recall  f1-score   support

           0       0.78      0.68      0.72       153
           1       0.85      0.90      0.87       303

    accuracy                           0.83       456
   macro avg       0.81      0.79      0.80       456
weighted avg       0.82      0.83      0.82       456
```





1.5.2.2 Naive Bayes (Model Tuning)

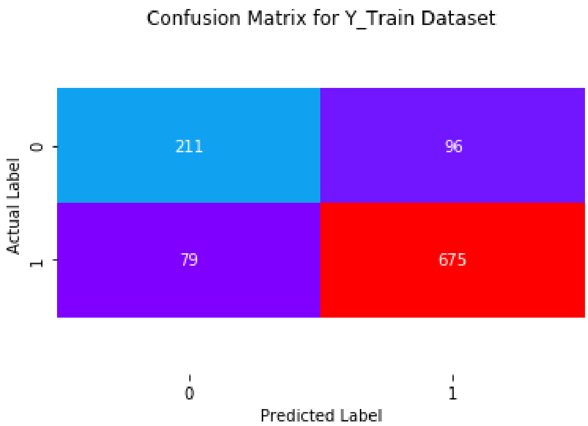
```
clf=GaussianNB()

nb_param={'var_smoothing':[1e-10,1e-09,1e-08,1e-07,1e-06,1e-05]}

nb_model=GridSearchCV(clf,param_grid=nb_param,n_jobs=2,cv=10)
nb_model.fit(X_train,y_train)
```

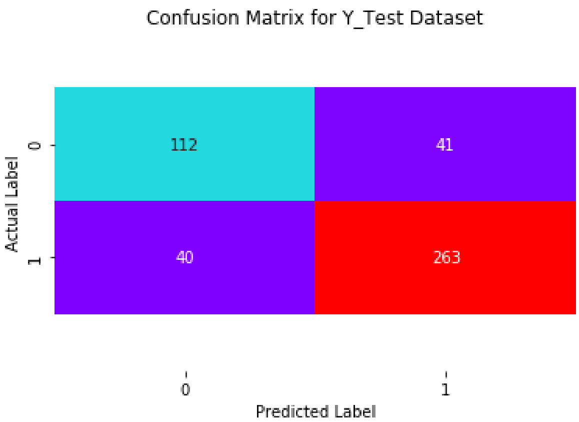
```
GridSearchCV(cv=10, estimator=GaussianNB(), n_jobs=2,
             param_grid={'var_smoothing': [1e-10, 1e-09, 1e-08, 1e-07, 1e-06,
                                           1e-05]})
```

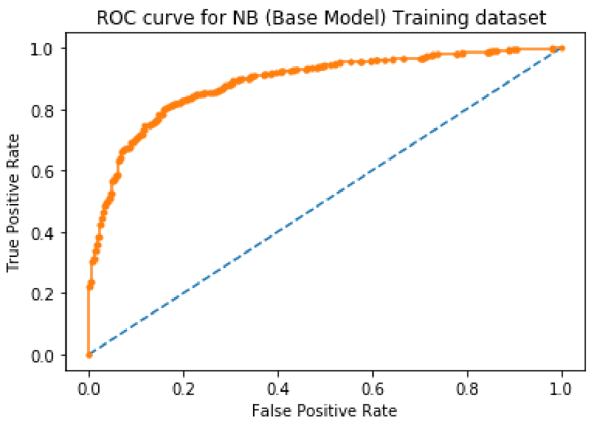
```
nb_model.best_params_
```

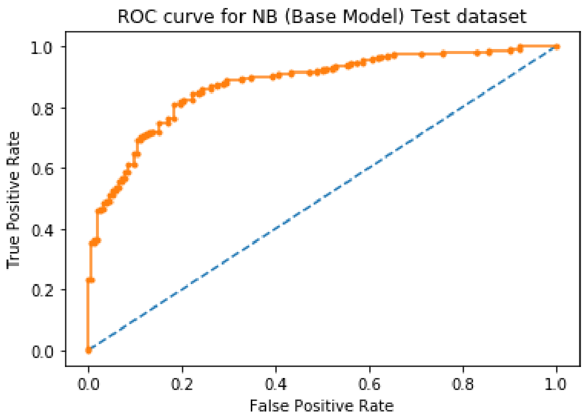
```
{'var_smoothing': 1e-10}
```

```
The accuracy score for the Training dataset is: 0.8350612629594723


The classification report for the Training dataset is:
              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


The accuracy score for the Testing dataset is: 0.8223684210526315


The classification report for the Testing dataset is:
              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
```









1.5.3.2 Support Vector Machine (Model Tuning)

```
from sklearn.svm import SVC

clf=SVC(probability=True)

sv_param={'C':[0.1,1.0,10,100],
          'tol':[0.001,0.01,0.1,1.0,10]}

sv_model=GridSearchCV(clf,param_grid=sv_param,cv=10)

sv_model.fit(X_train,y_train)
```

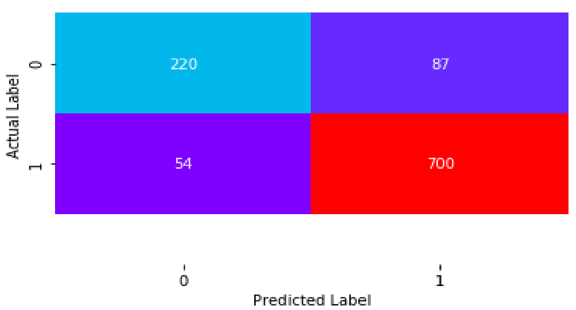
```
GridSearchCV(cv=10, estimator=SVC(probability=True),
             param_grid={'C': [0.1, 1.0, 10, 100],
                         'tol': [0.001, 0.01, 0.1, 1.0, 10]})
```

```
sv_model.best_params_
sv_grid=sv_model.best_estimator_
sv_grid
```

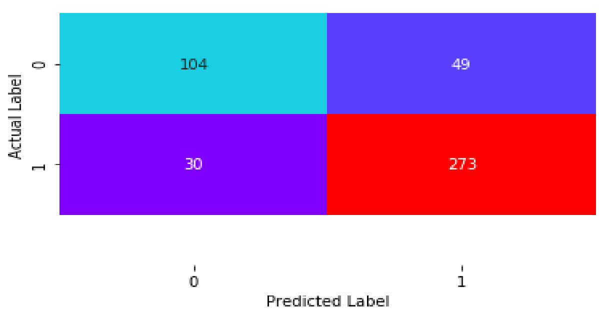
```
SVC(probability=True, tol=1.0)
```

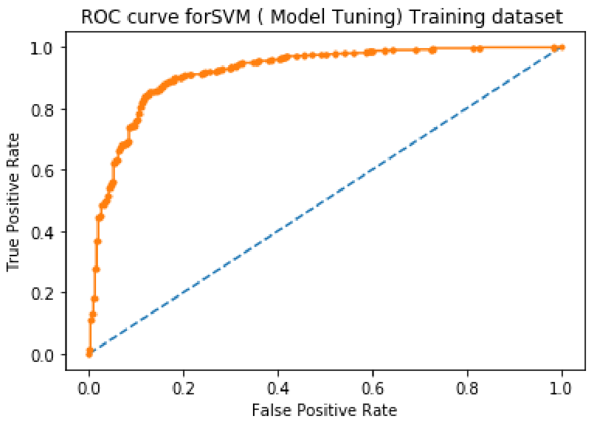
```
The accuracy score for the Training dataset is: 0.8671065032987747


The classification report for the Training dataset is:
              precision    recall  f1-score   support

           0       0.80      0.72      0.76       307
           1       0.89      0.93      0.91       754

    accuracy                           0.87      1061
   macro avg       0.85      0.82      0.83      1061
weighted avg       0.86      0.87      0.86      1061



The accuracy score for the Testing dataset is: 0.8267543859649122


The classification report for the Testing dataset is:
              precision    recall  f1-score   support

           0       0.78      0.68      0.72       153
           1       0.85      0.90      0.87       303

    accuracy                           0.83       456
   macro avg       0.81      0.79      0.80       456
weighted avg       0.82      0.83      0.82       456
```

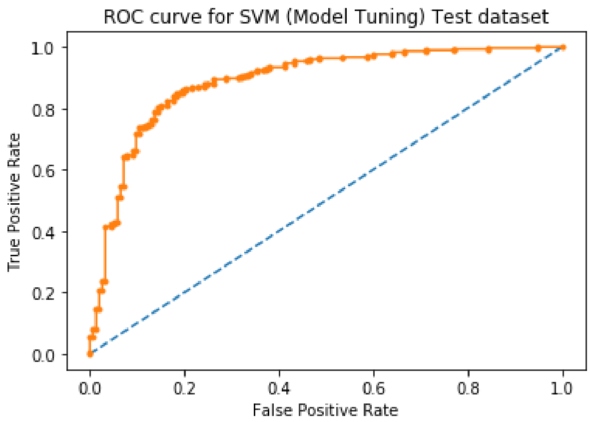
Confusion Matrix for Y_Train Dataset



Confusion Matrix for Y_Test Dataset







1.7.1 Ensemble Technique (Bagging -Decision Tree Used)

```
from sklearn.tree import DecisionTreeClassifier

dt=DecisionTreeClassifier(criterion='gini', max_depth=4,random_state=1)

dt.fit(X_train,y_train)
```

```
DecisionTreeClassifier(max_depth=4, random_state=1)
```

```
print(dt.score(X_train,y_train))
print(dt.score(X_test,y_test))
```

```
0.8444863336475024
0.7894736842105263
```

```
print(pd.DataFrame(dt.feature_importances_,columns=['IMP'],index=X_train.columns))
```

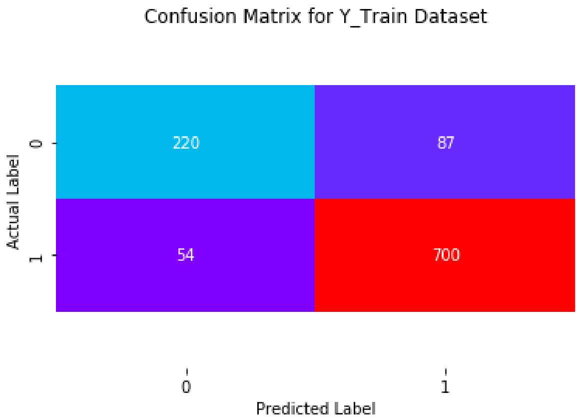
```
                              IMP
age                      0.042957
economic_cond_national   0.063725
economic_cond_household  0.013549
Blair                    0.246707
Hague                    0.435266
Europe                   0.136791
political_knowledge      0.061006
gender_male              0.000000
```

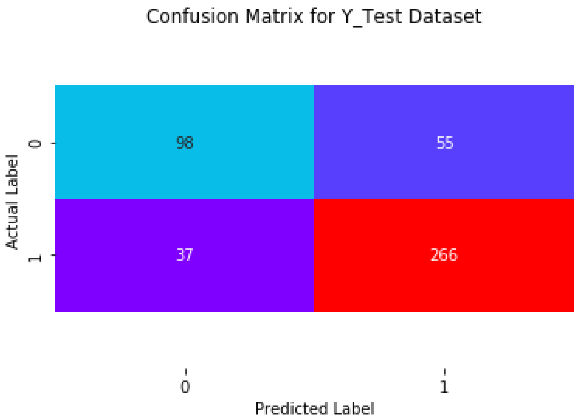
```
from sklearn.ensemble import BaggingClassifier

bc=BaggingClassifier(base_estimator=dt,bootstrap=True,random_state=1)

bc.fit(X_train,y_train)
```

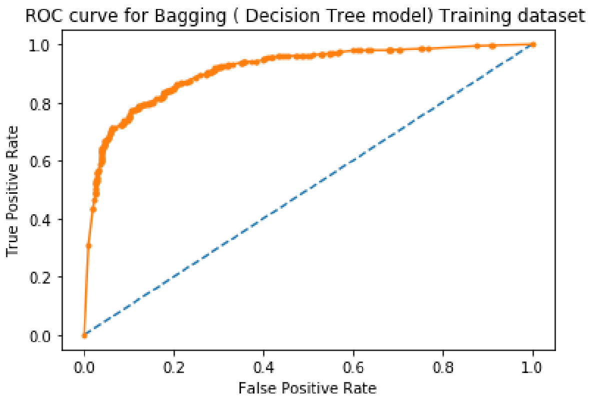
```
BaggingClassifier(base_estimator=DecisionTreeClassifier(max_depth=4,
                                                        random_state=1),
                  random_state=1)
```

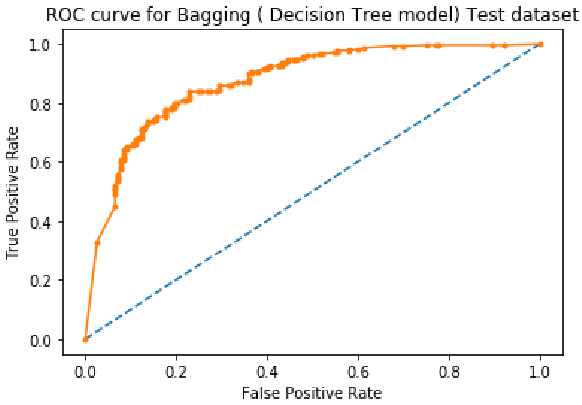
```
dcytrain_predict=bc.predict(X_train)
ytest_predict=bc.predict(X_test)
```

```
The accuracy score for the Training dataset is: 0.8586239396795476


The classification report for the Training dataset is:
              precision    recall  f1-score   support

           0       0.80      0.72      0.76       307
           1       0.89      0.93      0.91       754

    accuracy                           0.87      1061
   macro avg       0.85      0.82      0.83      1061
weighted avg       0.86      0.87      0.86      1061



The accuracy score for the Testing dataset is: 0.7982456140350878


The classification report for the Testing dataset is:
              precision    recall  f1-score   support

           0       0.73      0.64      0.68       153
           1       0.83      0.88      0.85       303

    accuracy                           0.80       456
   macro avg       0.78      0.76      0.77       456
weighted avg       0.79      0.80      0.79       456
```









1.7.2 Bagging Classifier (Base Estimator: Random Forest)

```python
from sklearn.ensemble import BaggingClassifier,RandomForestClassifier

rf_param = {'n_estimators':[50,100,150],'max_depth':[7,8,9,10],
    'min_samples_split':[50,70,90],'min_samples_leaf':[8,12,16],
    'max_features':[5,6,7]}

rfcl=RandomForestClassifier()

rf_grid=GridSearchCV(estimator=rfcl, param_grid=rf_param,cv=10)

rf_grid.fit(X_train,y_train)
```

```
GridSearchCV(cv=10, estimator=RandomForestClassifier(),
             param_grid={'max_depth': [7, 8, 9, 10], 'max_features': [5, 6, 7],
                         'min_samples_leaf': [8, 12, 16],
                         'min_samples_split': [50, 70, 90],
                         'n_estimators': [50, 100, 150]})
```

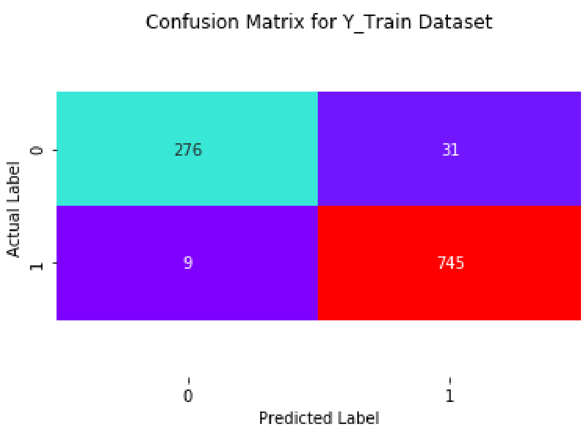
```python
rf_grid.best_params_
```

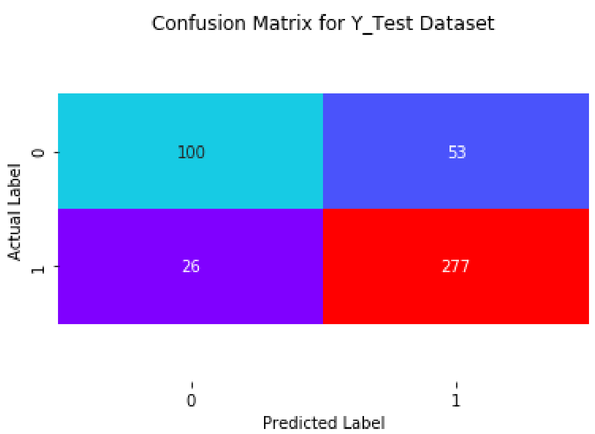
```
{'max_depth': 8,
 'max_features': 5,
 'min_samples_leaf': 12,
 'min_samples_split': 50,
 'n_estimators': 150}
```

```python
rf_model=rf_grid.best_estimator_
```

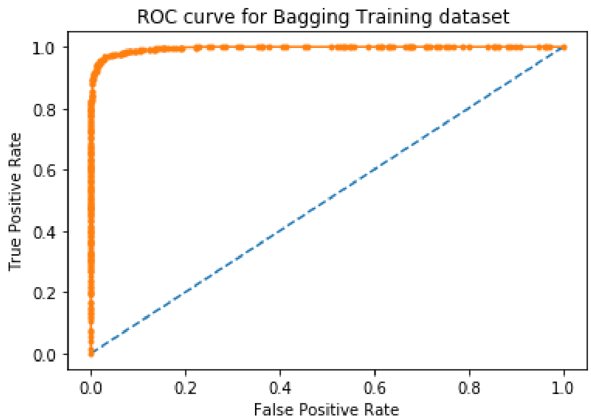
```python
bag=BaggingClassifier(base_estimator=rf_grid,n_estimators=10,random_state=1,)

bag.fit(X_train,y_train)
```

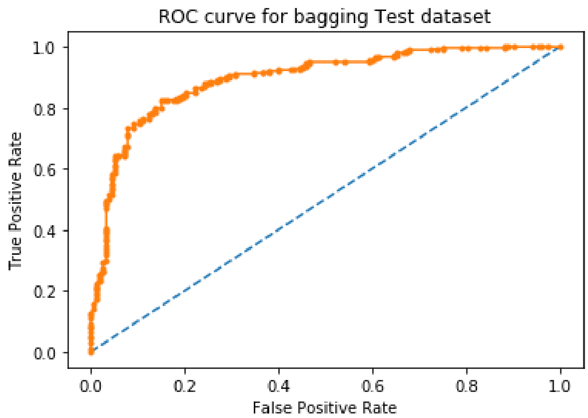
```python
ytrain_predict=bag.predict(X_train)
ytest_predict=bag.predict(X_test)
```





```
The accuracy score for the Training dataset is: 0.9622997172478793


The classification report for the Training dataset is:
              precision    recall  f1-score   support

           0       0.97      0.90      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


The accuracy score for the Testing dataset is: 0.8267543859649122


The classification report for the Testing dataset is:
              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
```





From the ROC curve we can see that the model is OVERFITTED.

1.7.2 Ada Boosting (Model Tuning)

```
from sklearn.ensemble import AdaBoostClassifier

ada=AdaBoostClassifier()

ada_param={'n_estimators':[300,500,1000],'learning_rate':[0.01,0.1,1]}

ada_grid=GridSearchCV(estimator=ada,param_grid=ada_param,cv=10)

ada_grid.fit(X_train,y_train)
```

```
GridSearchCV(cv=10, estimator=AdaBoostClassifier(),
             param_grid={'learning_rate': [0.01, 0.1, 1],
                         'n_estimators': [300, 500, 1000]})
```

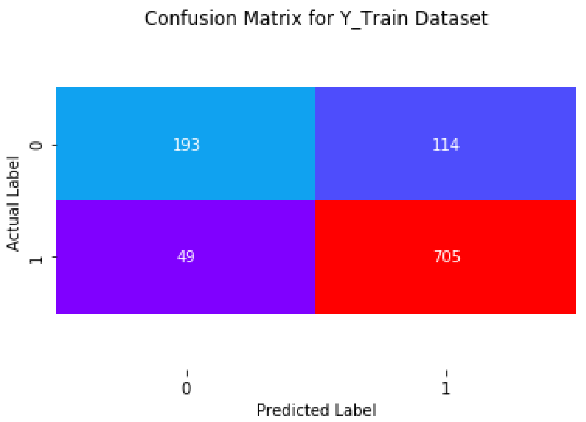
```
ada_grid.best_params_
```

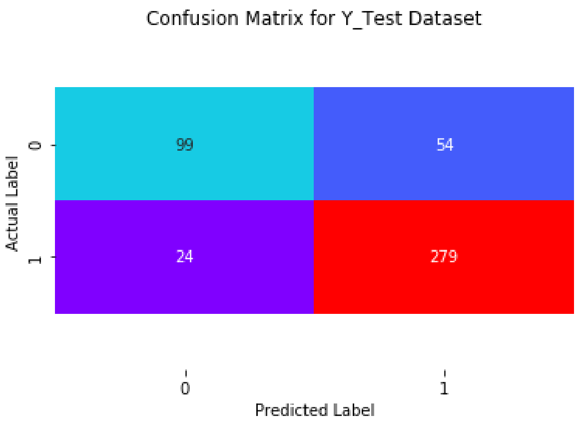
```
{'learning_rate': 0.1, 'n_estimators': 500}
```

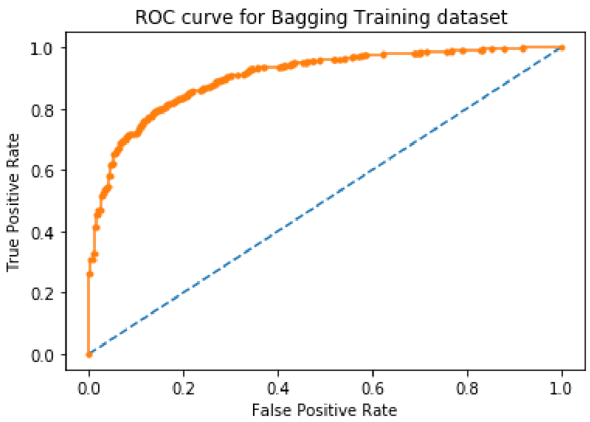
```
ada_model=ada_grid.best_estimator_
```

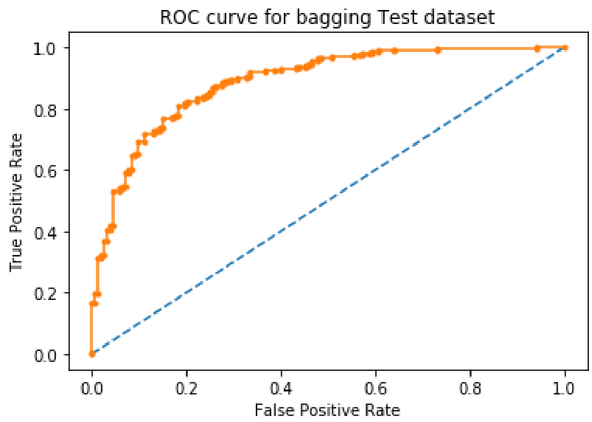
```
ytrain_predict=bag.predict(X_train)
ytest_predict=bag.predict(X_test)
```

```
The accuracy score for the Training dataset is: 0.8463713477851084


The classification report for the Training dataset is:
              precision    recall  f1-score   support

           0       0.80      0.63      0.70       307
           1       0.86      0.94      0.90       754

    accuracy                           0.85      1061
   macro avg       0.83      0.78      0.80      1061
weighted avg       0.84      0.85      0.84      1061



The accuracy score for the Testing dataset is: 0.8289473684210527


The classification report for the Testing dataset is:
              precision    recall  f1-score   support

           0       0.80      0.65      0.72       153
           1       0.84      0.92      0.88       303

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









1.7.2.2 Gradient Boosting (Model Tuning)

```python
from sklearn.ensemble import GradientBoostingClassifier

gbc=GradientBoostingClassifier()

gbc_param={'learning_rate':[0.0001,0.001,0.01,0.1,1],
           'n_estimators':[100,500]}

gb_grid=GridSearchCV(estimator=gbc,param_grid=gbc_param,cv=5)

gb_grid.fit(X_train,y_train)
```

```
GridSearchCV(cv=5, estimator=GradientBoostingClassifier(),
             param_grid={'learning_rate': [0.0001, 0.001, 0.01, 0.1, 1],
                         'n_estimators': [100, 500]})
```

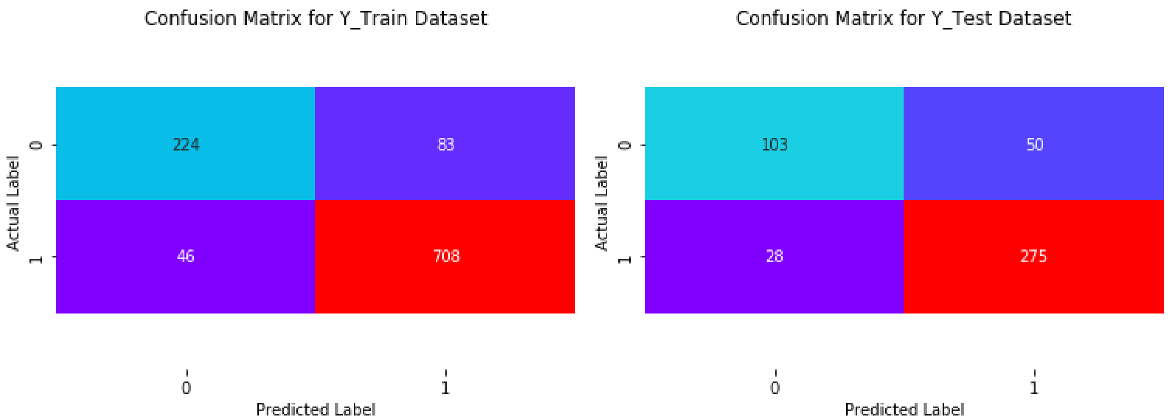
```python
gb_grid.best_params_
```

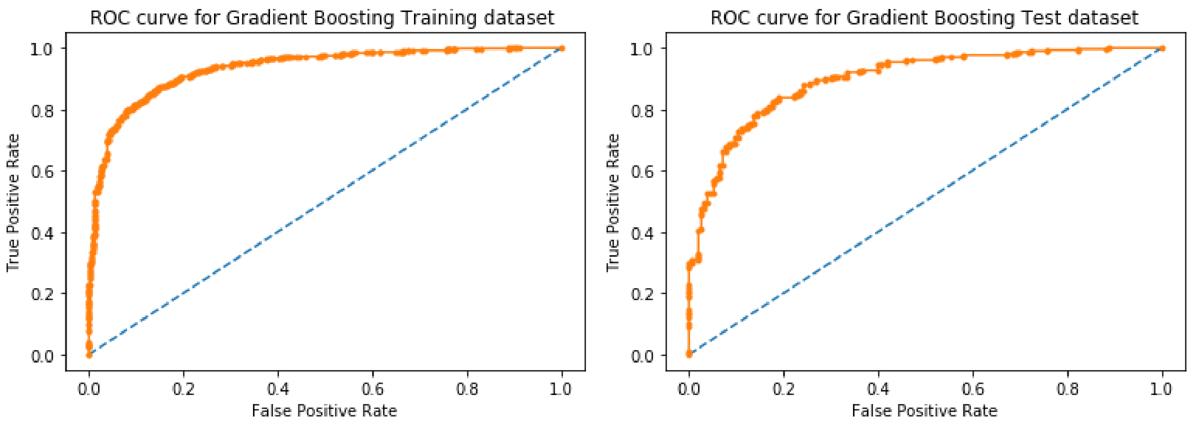
```
{'learning_rate': 0.01, 'n_estimators': 500}
```

```python
gb_model=gb_grid.best_estimator_
```

```python
ytrain_predict=gb_model.predict(X_train)
ytest_predict=gb_model.predict(X_test)
```

```
The accuracy score for the Training dataset is: 0.8784165881244109


The classification report for the Training dataset is:
              precision    recall  f1-score   support

           0       0.83      0.73      0.78       307
           1       0.90      0.94      0.92       754

    accuracy                           0.88      1061
   macro avg       0.86      0.83      0.85      1061
weighted avg       0.88      0.88      0.88      1061



The accuracy score for the Testing dataset is: 0.8289473684210527


The classification report for the Testing dataset is:
              precision    recall  f1-score   support

           0       0.79      0.67      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
```





1.7.2.3 Xtreme Gradient Boosting (Model Tuning)

```
import xgboost as xgb
```

```
clf=xgb.XGBClassifier(objective='binary:logistic',use_label_encoder=True,)

xgb_param={'C':[0.1,1,10,100,1000],'gamma':[0.1,0.01,0.001,0.0001]}

xgb_grid=GridSearchCV(estimator=clf, param_grid=xgb_param, cv=5)

xgb_grid.fit(X_train,y_train)
```

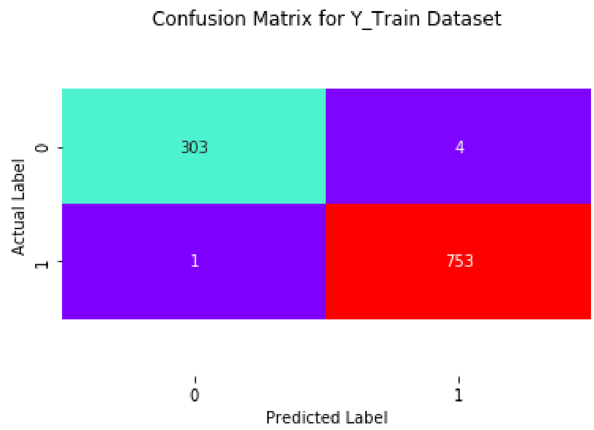
```
xgb_grid.best_params_
```

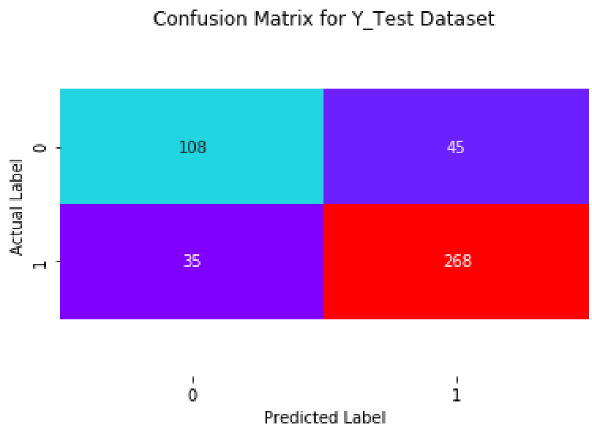
```
{'C': 0.1, 'gamma': 0.1}
```

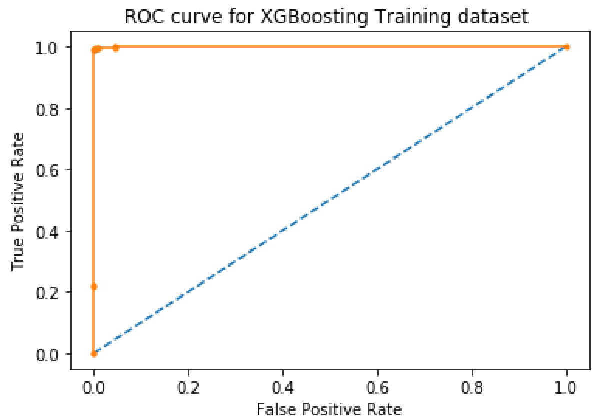
```
xgb_model=xgb_grid.best_estimator_
xgb_model
```

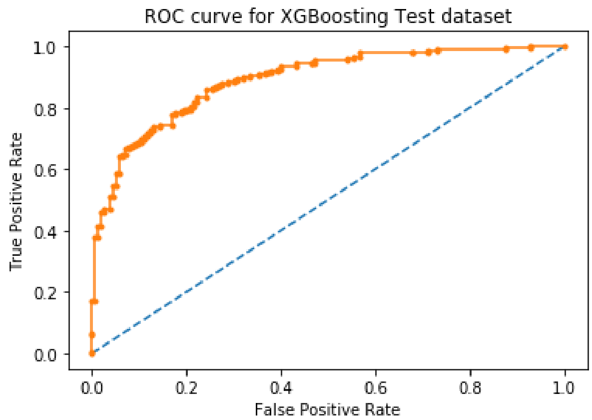
```
XGBClassifier(C=0.1, base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0.1, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=4, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

```
The accuracy score for the Training dataset is: 0.9952874646559849


The classification report for the Training dataset is:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       307
           1       0.99      1.00      1.00       754

    accuracy                           1.00      1061
   macro avg       1.00      0.99      0.99      1061
weighted avg       1.00      1.00      1.00      1061



The accuracy score for the Testing dataset is: 0.8245614035087719


The classification report for the Testing dataset is:
              precision    recall  f1-score   support

           0       0.76      0.71      0.73       153
           1       0.86      0.88      0.87       303

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









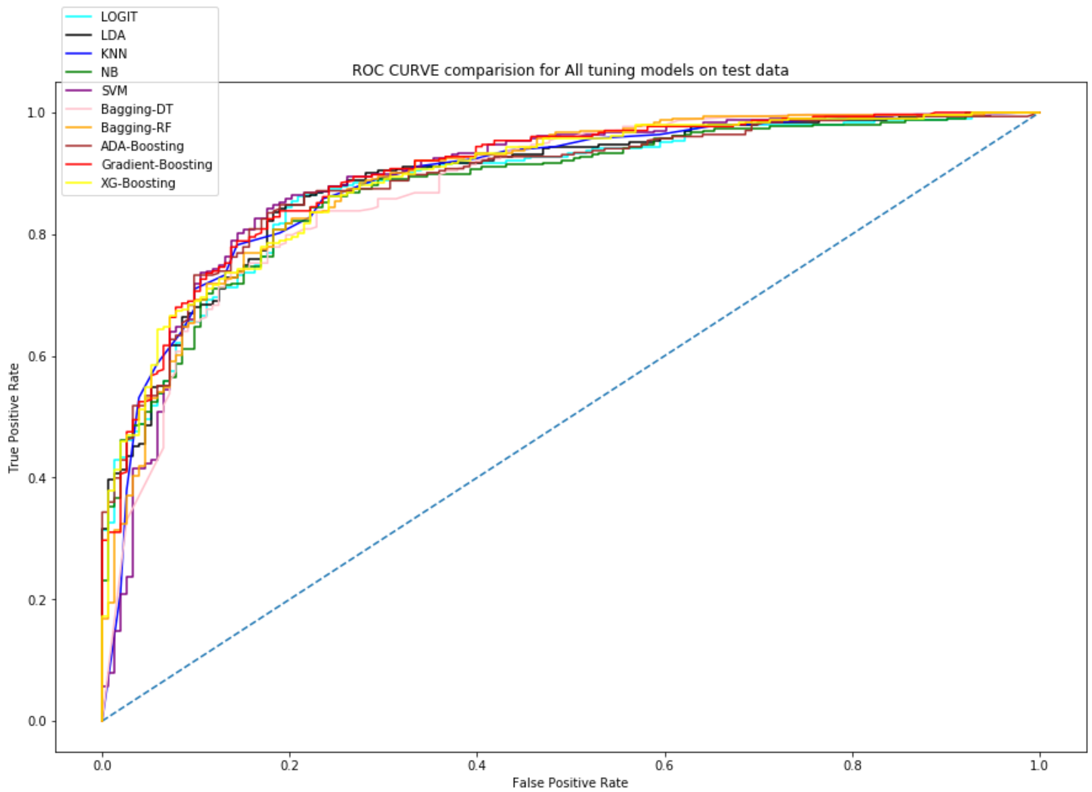
Inference from Model Tuning, Bagging, Boosting:

All the models have performed well in both the Training and Test data. While some model results between training and test sets having under or over fitting issues.

The output results for Labour Party as follows

- The Logistic Regression model performs well with good Precision, recall and f1 score.
- KNN model is better than LR model performs good train accuracy of 84%, but the test is 83%, which is also good. The precision, recall and f1 score is the same as Logistic regression.
- The SVM model performs well on both training and testing data set.
- Pruning/tuning DT with Gini index and max depth = 4, and the model performance is better And not overfitting with 84% train accuracy and 79% test accuracy.
- Applying Ada Boosting model and predicting the train and test. The train and test accuracy are 85% and 83% respecting
- Gradient Boosting model performs the best with 88% train accuracy and with 83% test accuracy. The precision, recall and f1 score is also good.
- RF model with bagging applied, performs similar to the normal RF as they are not different. The model has good recall,F1 and AUC score
- XGBoost is not performed well in testing data, OVERFITTED.

Model	Accuracy		Precision		Recall		F1 Score		AUC Score	
Labour	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Logit	0.84	0.84	0.86	0.86	0.93	0.89	0.89	0.87	0.889	0.882
LDA	0.83	0.84	0.86	0.87	0.91	0.88	0.89	0.88	0.889	0.888
KNN	0.84	0.83	0.87	0.85	0.91	0.90	0.89	0.87	0.899	0.886
NB	0.84	0.82	0.88	0.87	0.90	0.87	0.89	0.87	0.888	0.876
SVM	**0.87**	**0.83**	**0.89**	**0.85**	**0.93**	**0.90**	**0.91**	**0.87**	**0.919**	**0.889**
Bagging	0.87	0.80	0.89	0.83	0.93	0.88	0.91	0.85	0.910	0.874
Bagging-RF	0.96	0.83	0.96	0.84	0.99	0.91	0.97	0.88	0.905	0.889
Ada-Boost	0.85	0.83	0.86	0.84	0.94	0.92	0.90	0.88	0.911	0.888
Grad-Boost	**0.88**	**0.83**	**0.90**	**0.85**	**0.94**	**0.91**	**0.92**	**0.88**	**0.934**	**0.890**
XGBoost	1.00	0.82	0.99	0.86	1.00	0.88	1.00	0.87	1.00	0.891



1.8 Based on your analysis and working on the business problem, detail out appropriate insights and recommendations to help the management solve the business objective.

For Labour Party:

- As per survey data 69.70% Voters are predicted to vote for labour party.
- The accuracy of the model is the best metrics for this use case as true positive and true negative predictions are essential to predict the **exit poll**.
- The best-case scenario is based on **SVM** and **Gradient Boosting** model. The worst-case scenario is based on **XGB model (Overfit).**
- **Therefore, working out the best-case scenario, given the accuracy 84%, a minimum of 58.54% of seat share is assured for labour party.**
- **In the worst-case scenario, given the accuracy of 82%, a minimum of 57.15% of seat share is assured for labour party.**
- When the issues such as Euroscepticism, national & household economic condition and knowledge of the parties on European integration are considered as influential variables for the election, lesser predictability for the conservative party in all the predictions infers that there are many fence sitters who are expected to vote for the conservative party and likelihood of getting influenced by other issues.
- The labour party supporters spread across complete spectrum of age have been influenced to a certain extent by high positive perception about a strong national economic condition & household economic condition.
- Voter's strong scepticism of European integration have comparatively good influence on the voting pattern. However labour party's stand on European integration takes away voters to conservative party to certain extent.
- Out of the voters choosing labour party, 52% of supporters are female and 48% are male. Labour Party can try ways to attract Male supporters to increase vote bank.

For Conservative Party:

Model	Accuracy		Precision		Recall		F1 Score		AUC Score	
Labour	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Logit	0.84	0.83	0.79	0.76	0.63	0.71	0.70	0.73	0.843	0.826
LDA	0.83	0.84	0.74	0.76	0.65	0.74	0.69	0.75	0.833	0.835
KNN	0.84	0.83	0.75	0.78	0.66	0.68	0.70	0.72	0.837	0.826
NB	0.84	0.82	0.73	0.74	0.69	0.73	0.71	0.73	0.835	0.822
SVM	**0.87**	**0.83**	**0.80**	**0.78**	**0.72**	**0.68**	**0.76**	**0.72**	**0.867**	**0.826**
Bagging	0.87	0.80	0.80	0.73	0.72	0.64	0.76	0.68	0.858	0.798
Bagging-RF	0.96	0.83	0.97	0.79	0.90	0.65	0.93	0.72	0.962	0.826
Ada-Boost	0.85	0.83	0.80	0.80	0.63	0.65	0.70	0.72	0.846	0.828
Grad-Boost	**0.88**	**0.83**	**0.83**	**0.79**	**0.73**	**0.67**	**0.78**	**0.73**	**0.878**	**0.828**
XGBoost	1.00	0.82	1.00	0.76	0.99	0.71	0.99	0.73	1.000	0.891

- As per survey data 30.32% Voters are predicted to vote for Conservative party.
- The accuracy of the model is the best parameter for this use case as true positive and true negative predictions are essential to predict the exit poll.
- The best-case scenario is based on SVM and Gradient Boosting model. The worst-case scenario is based on XGB model.
- **Therefore, working out the best-case scenario, given the accuracy 83%, a minimum of 25.16% of seat share is assured for Conservative party.**
- **In the worst-case scenario, given the accuracy of 80%, a minimum of 24.25% of seat share is assured for Conservative party.**

PROBLEM 2: Text Mining

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

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

Importing libraries and dataset:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import nltk
import re

nltk.download('inaugural')
nltk.download('stopwords')
nltk.download('punkt')

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.corpus import inaugural
inaugural.fileids()
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
from wordcloud import WordCloud,STOPWORDS
from PIL import Image

```

NLTK will provide you with everything from splitting paragraphs to sentences, splitting words, identifying the part of speech, highlighting themes, and even helping your machine understand what the text is about.

Loading the text file in Pandas Dataframe:

The below data frame is the speeches of the Presidents of the United States of America

	president	speech
0	Franklin D. Roosevelt	On each national day of inauguration since 178...
1	John F. Kennedy	Vice President Johnson, Mr. Speaker, Mr. Chief...
2	Richard Nixon	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...

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

Number of Characters present in the Speech:

Below we are counting the total number of words from each file separately.

Here we are using the .str.len() to count the total number of characters.

	president	speech	no_of_characters
0	Franklin D. Roosevelt	On each national day of inauguration since 178...	7571
1	John F. Kennedy	Vice President Johnson, Mr. Speaker, Mr. Chief...	7618
2	Richard Nixon	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	9991

Number of Words present in the Speech:

Below we are counting the total number of words from each file separately.

Here we are using the split() to split up the words based on space between each word and we are counting the total number of words by using the len() function

	president	speech	no_of_characters	no_of_words
0	Franklin D. Roosevelt	On each national day of inauguration since 178...	7571	1360
1	John F. Kennedy	Vice President Johnson, Mr. Speaker, Mr. Chief...	7618	1390
2	Richard Nixon	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	9991	1819

Number of Tokens present in the Speech:

	president	speech	no_of_characters	no_of_words	no_of_tokens
0	Franklin D. Roosevelt	On each national day of inauguration since 178...	7571	1360	1526
1	John F. Kennedy	Vice President Johnson, Mr. Speaker, Mr. Chief...	7618	1390	1543
2	Richard Nixon	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	9991	1819	2006

Number of Sentence present in the Speech:

Below we are counting the total number of sentence in each text file, by using lambda function. We are using pd.Dataframe to move the data as dictionary and then with lambda function we are checking each sentece which ends with "." Using endswith() function and the below code and output is as below.

	president	speech	no_of_characters	no_of_words	no_of_tokens	no_of_sentence
0	Franklin D. Roosevelt	On each national day of inauguration since 178...	7571	1360	1526	67
1	John F. Kennedy	Vice President Johnson, Mr. Speaker, Mr. Chief...	7618	1390	1543	52
2	Richard Nixon	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	9991	1819	2006	68

2.2) Remove all the stopwords from the three speeches.

```
from nltk.corpus import stopwords
stop=stopwords.words('english')
print('The common stopwords present in english are',stop[:9],'etc.,')
```

```
The common stopwords present in english are ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you'] etc.,
```

Removing all the punctuations, symbols, extra spaces:

	president	speech	processed_speech
0	Franklin D. Roosevelt	On each national day of inauguration since 178...	On each national day of inauguration since 178...
1	John F. Kennedy	Vice President Johnson, Mr. Speaker, Mr. Chief...	Vice President Johnson Mr Speaker Mr Chief Jus...
2	Richard Nixon	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	Mr Vice President Mr Speaker Mr Chief Justice ...

Converting all words to Lower case to make unique:

	president	speech	processed_speech
0	Franklin D. Roosevelt	On each national day of inauguration since 178...	on each national day of inauguration since 178...
1	John F. Kennedy	Vice President Johnson, Mr. Speaker, Mr. Chief...	vice president johnson mr speaker mr chief jus...
2	Richard Nixon	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	mr vice president mr speaker mr chief justice ...

The total number of stop words:

	president	speech	processed_speech	no_of_stopwords
0	Franklin D. Roosevelt	On each national day of inauguration since 178...	on each national day of inauguration since 178...	711
1	John F. Kennedy	Vice President Johnson, Mr. Speaker, Mr. Chief...	vice president johnson mr speaker mr chief jus...	672
2	Richard Nixon	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	mr vice president mr speaker mr chief justice ...	969

The total number words after removal of stop words:

	president	speech	processed_speech	no_of_stopwords	no_of_words_after_removal_of_stopwords
0	Franklin D. Roosevelt	On each national day of inauguration since 178...	national day inauguration since 1789 people re...	711	627
1	John F. Kennedy	Vice President Johnson, Mr. Speaker, Mr. Chief...	vice president johnson mr speaker mr chief jus...	672	693
2	Richard Nixon	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	mr vice president mr speaker mr chief justice ...	969	833

```
df1['processed_speech']
```

```
0    national day inauguration since 1789 people re...
1    vice president johnson mr speaker mr chief jus...
2    mr vice president mr speaker mr chief justice ...
Name: processed_speech, dtype: object
```

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 stop words)

```
roosvelt_freq = pd.Series(' '.join(df1['processed_speech'][[0]]).split()).value_counts()[:3]

kennedy_freq = pd.Series(' '.join(df1['processed_speech'][[1]]).split()).value_counts()[:3]

nixon_freq = pd.Series(' '.join(df1['processed_speech'][[2]]).split()).value_counts()[:3]
```

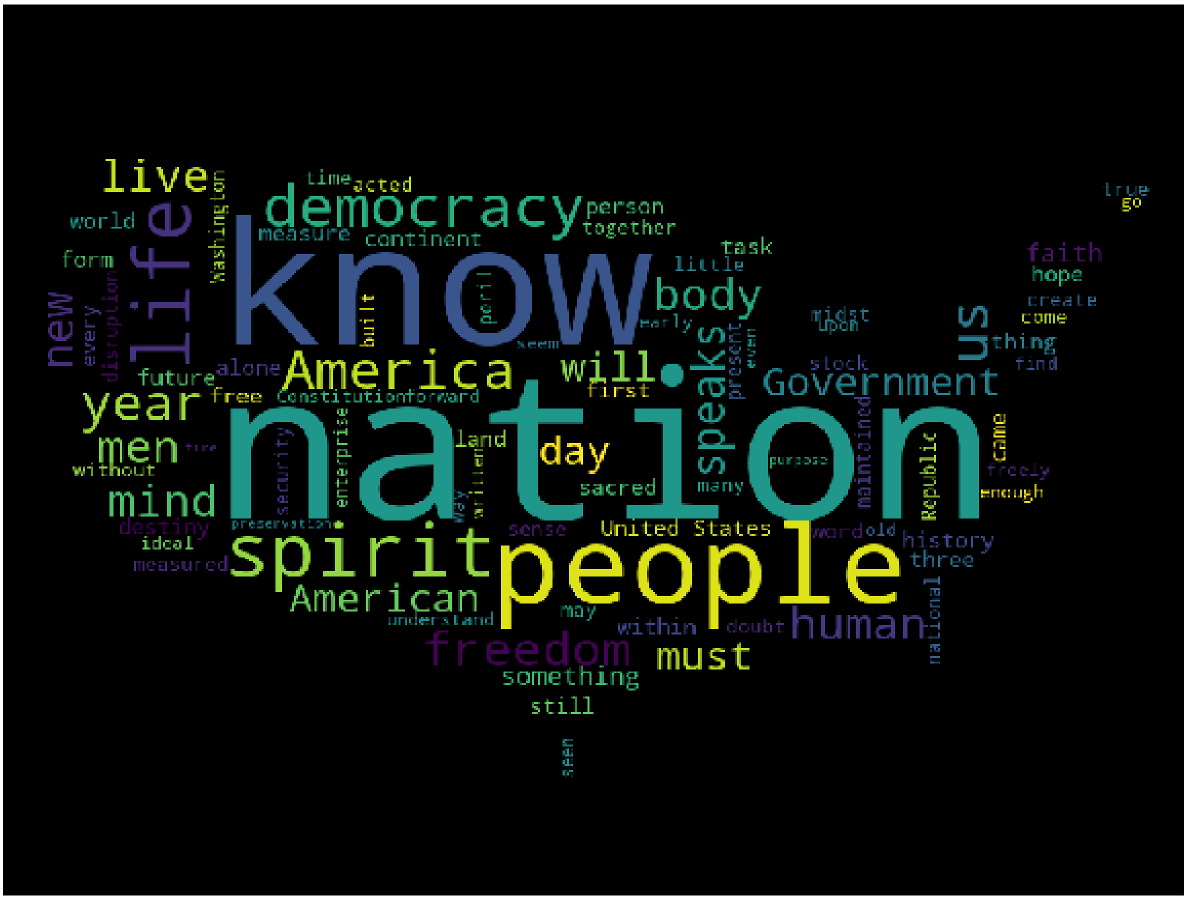
```
The top three words occur most of the time in president Franklin D. Roosevelt speech are:
 nation    11
know      10
spirit     9
dtype: int64


The top three words occur most of the time in president John F. Kennedy speech are:
 let       16
us        12
world      8
dtype: int64


The top three words occur most of the time in president Richard Nixon speech are:
 us        26
let       22
peace     19
dtype: int64
```

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

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. Word clouds are widely used for analysing data from social network websites.

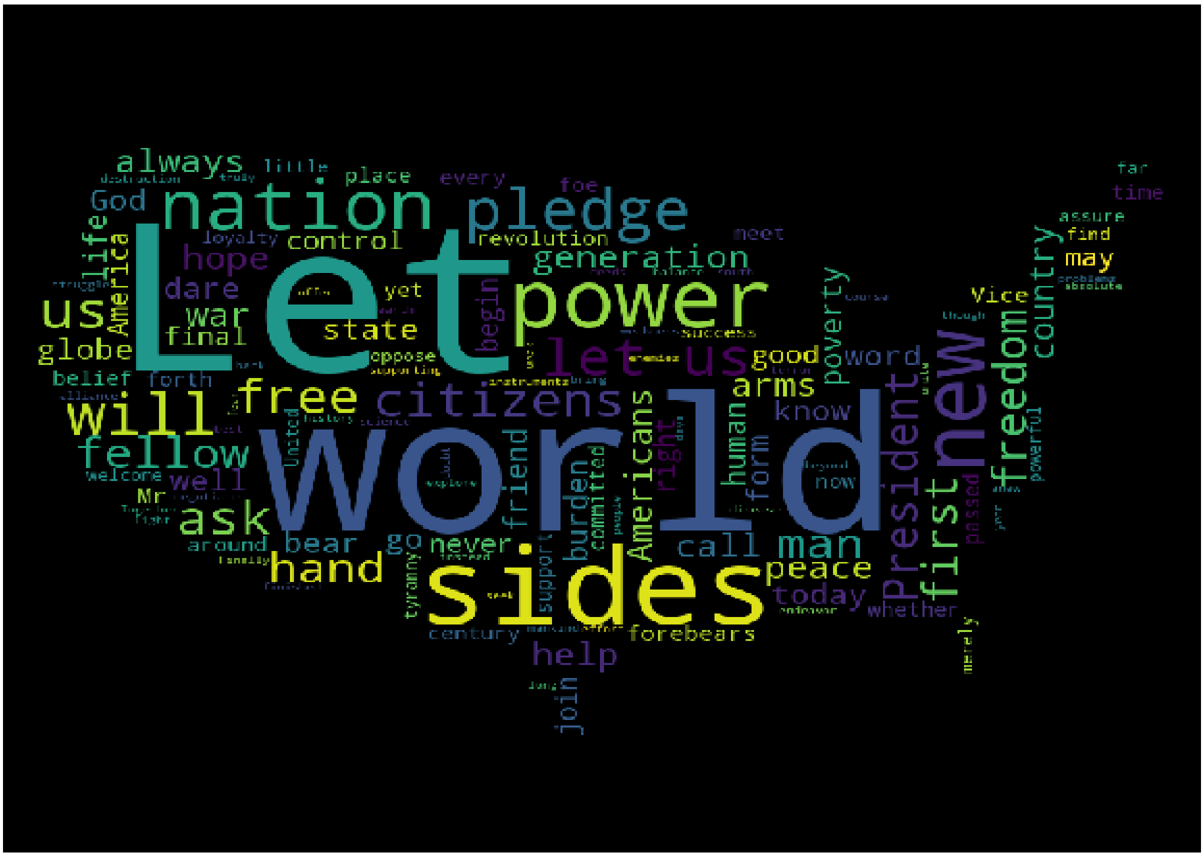
```
rs = df['speech'][[0]].apply(lambda x: ' '.join([x for x in x.split()]))
na_r = ' '.join(rs)

mask_r = np.array(Image.open(r'C:\Users\JAI\OneDrive\Desktop\america.jpg'))

wordcloud = WordCloud(width = 6000, height = 4000, background_color ='black',
                      min_font_size = 10, random_state=100,mask=mask_r).generate(na_r)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.xlabel('Word Cloud')
plt.tight_layout(pad = 0)

print("Word Cloud for president Franklin D. Roosevelt (Before cleaning)!!")
```

Word Cloud for president Franklin D. Roosevelt (Before cleaning):

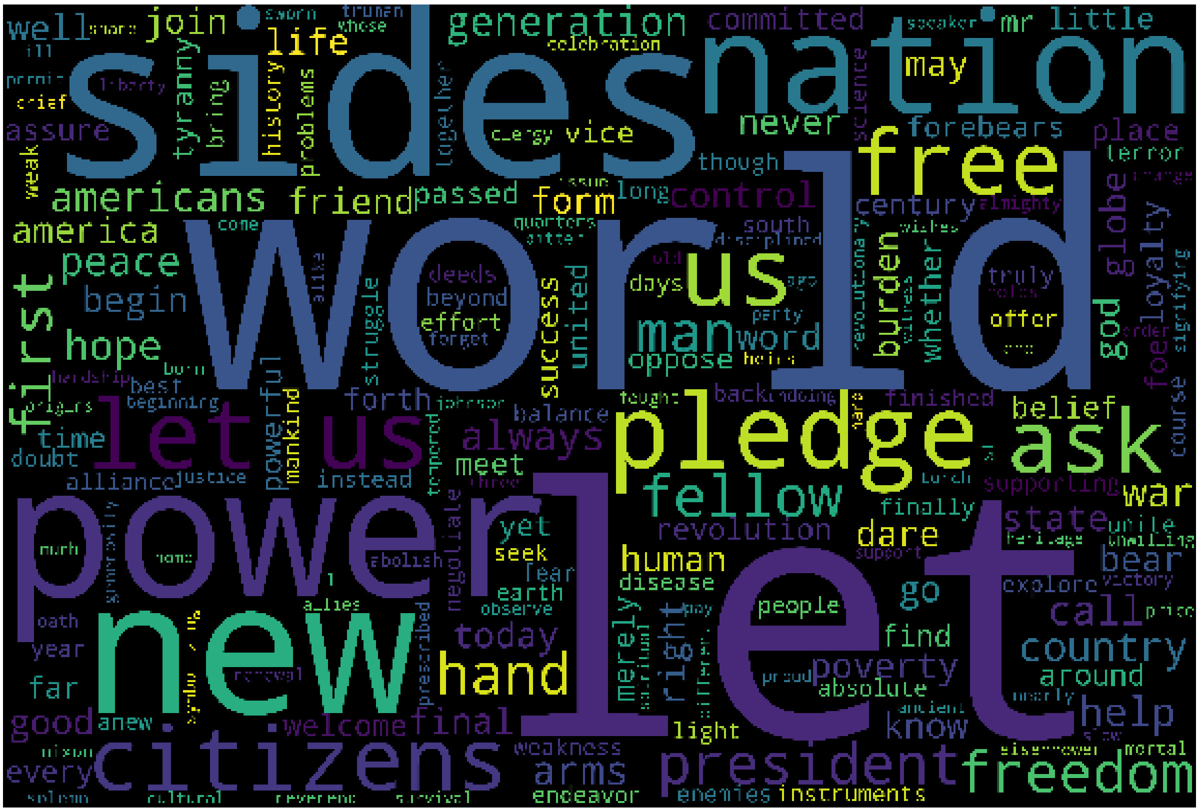


Word Cloud for president Franklin D. Roosevelt (After cleaning):

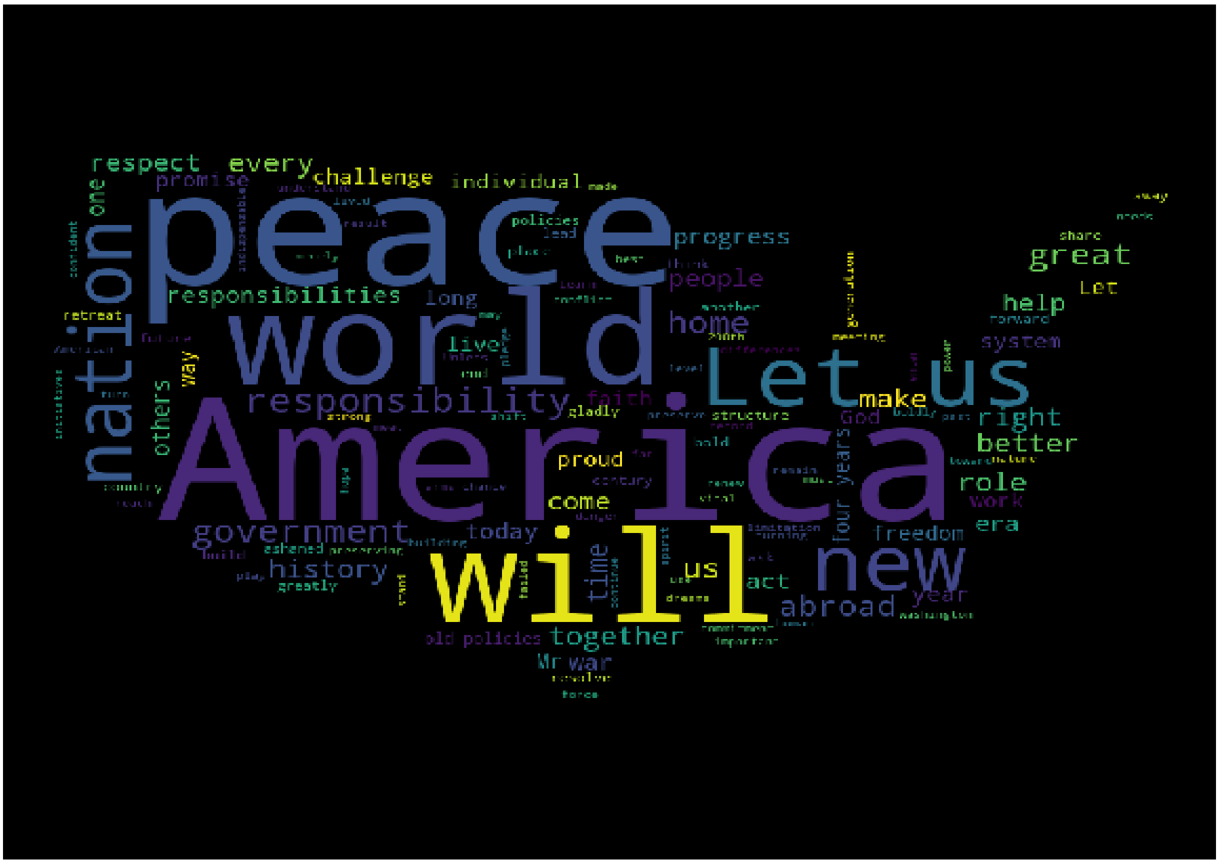
Word Cloud for president John F Kennedy (Before cleaning):



Word Cloud for president John F Kennedy (After cleaning):



Word Cloud for president Richard Nixon (Before cleaning):



Word Cloud for president Richard Nixon (After cleaning):