greatlearning
Power Ahead

Machine Learning

20th Feb 2022
Data Science and Business Analytics
(PGP-DSBA)
Online September_A 2021-22

Made By:
Anmol Tripathi

Table of Contents:

Problem 1:

Problem 2:

Problem 1:

Executive Summary:

The given data set has number columns and rows in them. In this data set we are given dataset of Election. Here in the data, voting of every individual is being mentioned and the person needs to work upon the data accordingly.

Background of the problem:

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

Data Description:

The shape of the data set seems to be with 26967 rows and 11 columns.

```
data_df.shape
```

```
(1525, 9)
```

- No null values in the dataset.
- The Data set is having int and object Data types.
- There are duplicate values present in the data set.

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

```
Total no of duplicate values = 8
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   vote                    1525 non-null   object
 1   age                     1525 non-null   int64
 2   economic.cond.national  1525 non-null   int64
 3   economic.cond.household 1525 non-null   int64
 4   Blair                   1525 non-null   int64
 5   Hague                   1525 non-null   int64
 6   Europe                  1525 non-null   int64
 7   political.knowledge     1525 non-null   int64
 8   gender                  1525 non-null   object
dtypes: int64(7), object(2)
memory usage: 107.4+ KB
```

Sample Data Set:

	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

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
1520	Conservative	67	5	3	2	4	11	3	male
1521	Conservative	73	2	2	4	4	8	2	male
1522	Labour	37	3	3	5	4	2	2	male
1523	Conservative	61	3	3	1	4	11	2	male
1524	Conservative	74	2	3	2	4	11	0	female

The above table represents the actual representation of the data set.

1.1 Read the dataset. Describe the data briefly. Interpret the inferences for each. Initial steps like head() .info(), Data Types, etc . Null value check, Summary stats, Skewness must be discussed.

Sample Data set:

	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

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
1520	Conservative	67	5	3	2	4	11	3	male
1521	Conservative	73	2	2	4	4	8	2	male
1522	Labour	37	3	3	5	4	2	2	male
1523	Conservative	61	3	3	1	4	11	2	male
1524	Conservative	74	2	3	2	4	11	0	female

Here “Unnamed:0” is a variable that simply represents the index of the data. Hence, it is dropped as it is of no use.

Data Description:

	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

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
dtype: int64
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   vote                    1525 non-null   object
 1   age                     1525 non-null   int64
 2   economic.cond.national  1525 non-null   int64
 3   economic.cond.household 1525 non-null   int64
 4   Blair                   1525 non-null   int64
 5   Hague                   1525 non-null   int64
 6   Europe                  1525 non-null   int64
 7   political.knowledge     1525 non-null   int64
 8   gender                  1525 non-null   object
dtypes: int64(7), object(2)
memory usage: 107.4+ KB
```

There are in total 10 variables present in the data set out of which 2 are categorical and 7 are of numeric type.
Also, there are no null values are present in the data set.

Duplicate Values:

```
Total no of duplicate values = 8
```

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
67	Labour	35	4	4	5	2	3	2	male
626	Labour	39	3	4	4	2	5	2	male
870	Labour	38	2	4	2	2	4	3	male
983	Conservative	74	4	3	2	4	8	2	female
1154	Conservative	53	3	4	2	2	6	0	female
1236	Labour	36	3	3	2	2	6	2	female
1244	Labour	29	4	4	4	2	2	2	female
1438	Labour	40	4	3	4	2	2	2	male

- There are in total 8 duplicate rows present in the dataset.
- Since, there is no identification or unique code for each row present.
- We cannot clearly say that this is the same person or different. So, we will not remove the duplicates in this case.

Shape of the Data Set:

```
data_df.shape
```

```
(1525, 9)
```

```
the no of rows  1525   the no of colums  9
```

Unique Values for Categorical Data Type:

```
Labour          1063
Conservative     462
Name: vote, dtype: int64
```

```
VOTE   2
Conservative     462
Labour          1063
Name: vote, dtype: int64
GENDER   2
male      713
female    812
Name: gender, dtype: int64
```

Data Type:

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

Skewness Value:

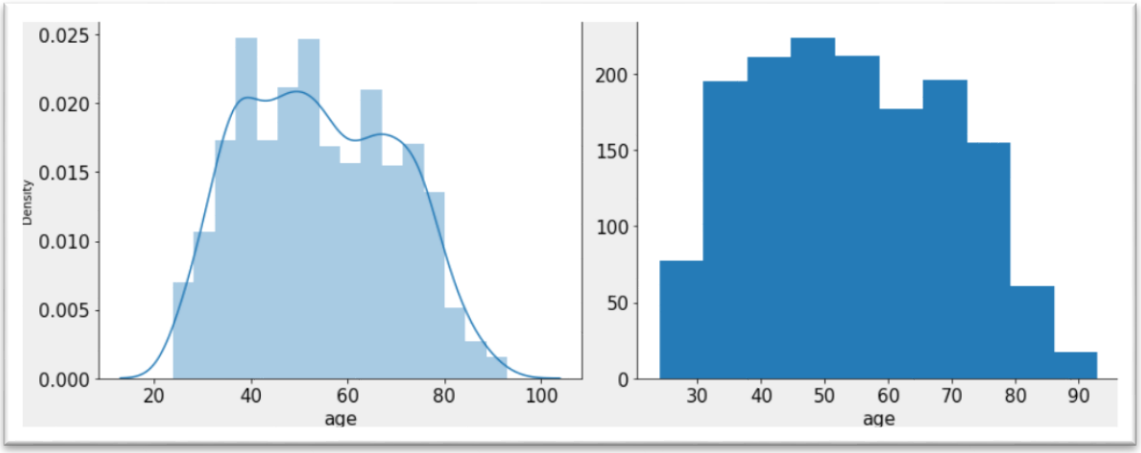
```
Hague                        0.152100
age                          0.144621
Europe                      -0.135947
economic.cond.household     -0.149552
economic.cond.national      -0.240453
political.knowledge         -0.426838
Blair                       -0.535419
dtype: float64
```

- Skewness is the measure of the asymmetry of the probability distribution of the real valued random variable about its mean.
- Only two variables are positively skewed and rest negatively skewed with max skewedness in Blair.

1.2) Perform EDA (Check the null values, Data types, shape, Univariate, bivariate analysis). Also check for outliers (4 pts). Interpret the inferences for each (3 pts) Distribution plots(histogram) or similar plots for the continuous columns. Box plots, Correlation plots. Appropriate plots for categorical variables. Inferences on each plot. Outliers proportion should be discussed, and inferences from above used plots should be there. There is no restriction on how the learner wishes to implement this but the code should be able to represent the correct output and inferences should be logical and correct.

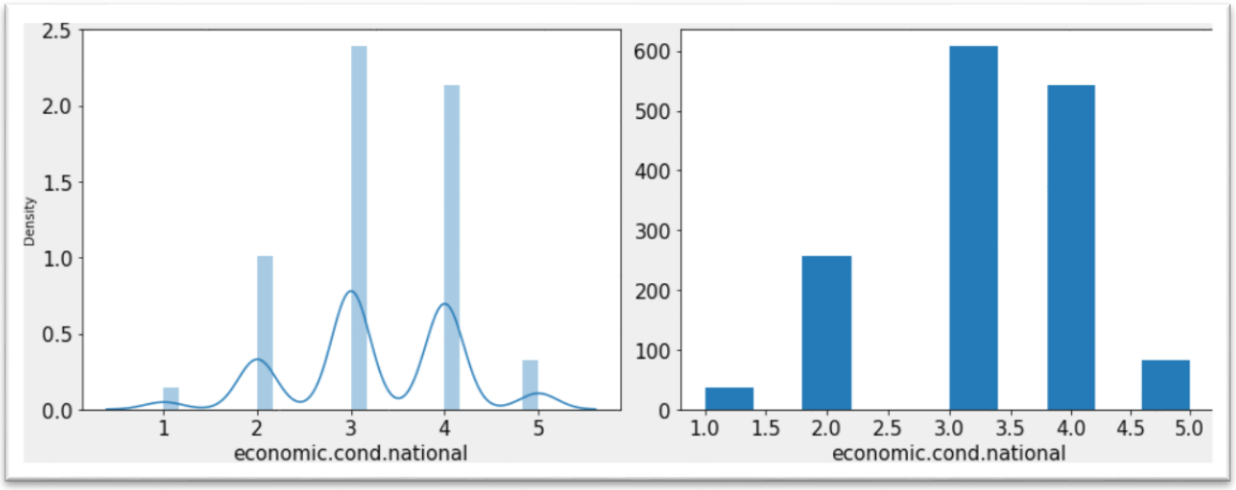
Age:

```
Range of values:  69
Minimum Age:  24
Maximum Age:  93
Mean value:  54.18229508196721
Median value:  53.0
Standard deviation:  15.711208571641977
Null values:  False
```



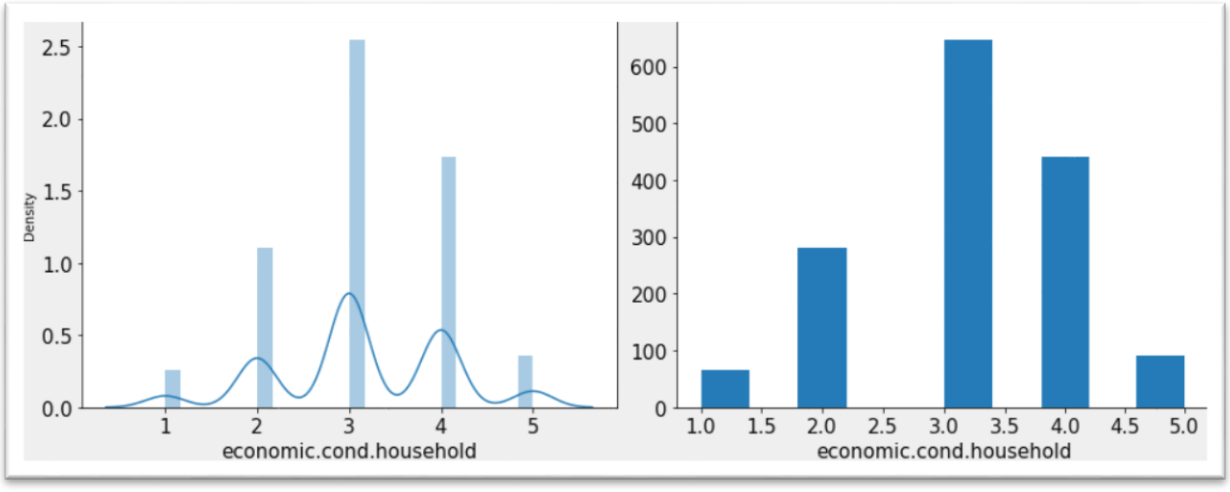
economic.cond.national:

```
Range of values:  4
Minimum Age:  1
Maximum Age:  5
Mean value:  3.2459016393442623
Median value:  3.0
Standard deviation:  0.8809692844149642
Null values:  False
```



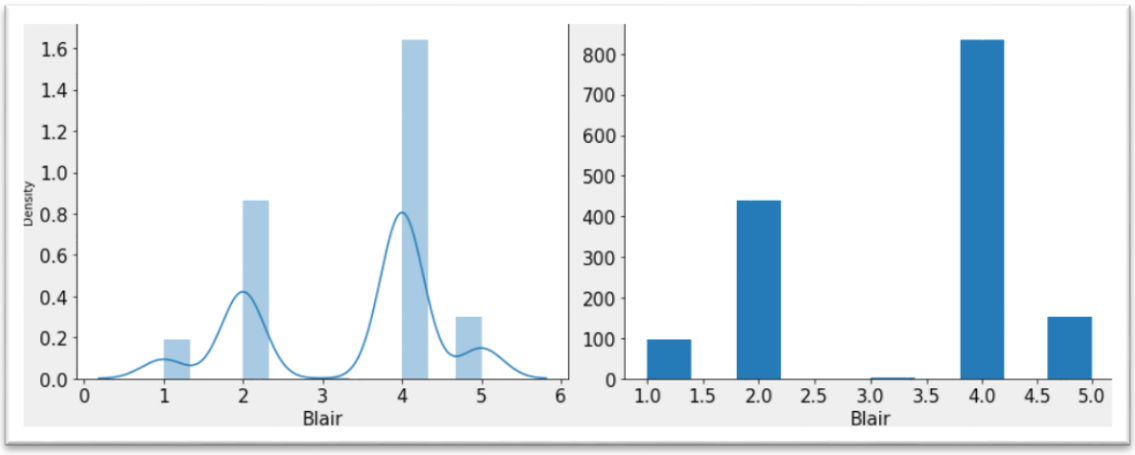
Economic.cond.household:

```
Range of values:  4
Minimum Age:  1
Maximum Age:  5
Mean value:  3.140327868852459
Median value:  3.0
Standard deviation:  0.9299513985782148
Null values:  False
```



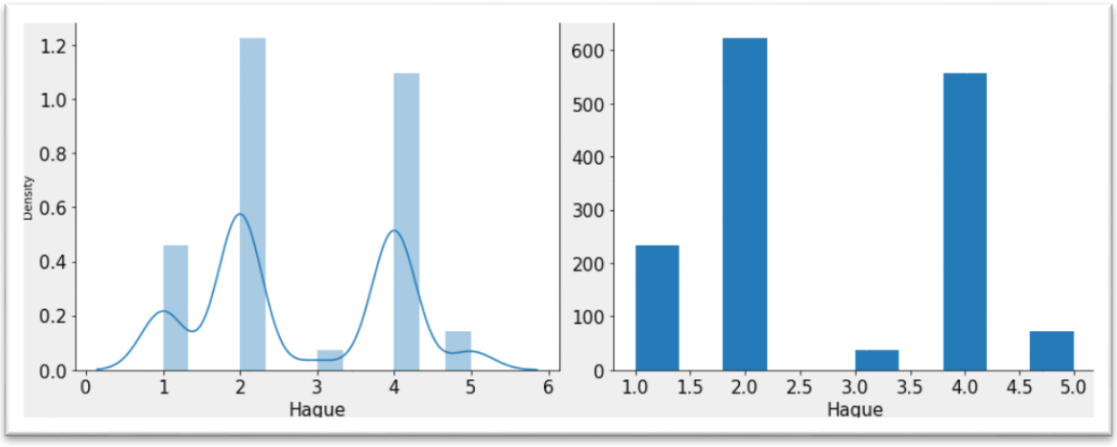
Blair:

```
Range of values:  4
Minimum Age:  1
Maximum Age:  5
Mean value:  3.3344262295081966
Median value:  4.0
Standard deviation:  1.1748241123034677
Null values:  False
```



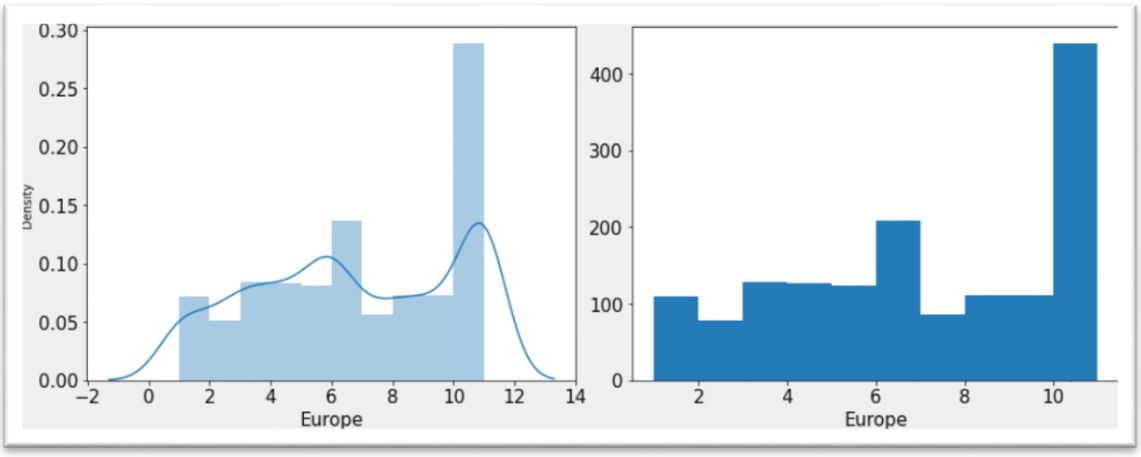
Hague:

```
Range of values:  4
Minimum Age:  1
Maximum Age:  5
Mean value:  2.7468852459016393
Median value:  2.0
Standard deviation:  1.2307034736168108
Null values:  False
```



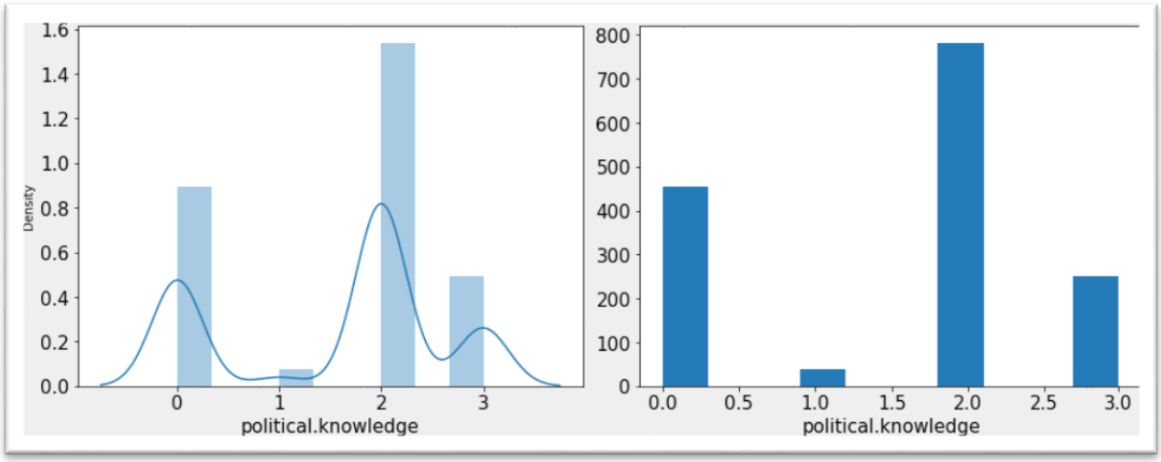
Europe:

```
Range of values:  10
Minimum Age:  1
Maximum Age:  11
Mean value:  6.728524590163935
Median value:  6.0
Standard deviation:  3.297538370463229
Null values:  False
```

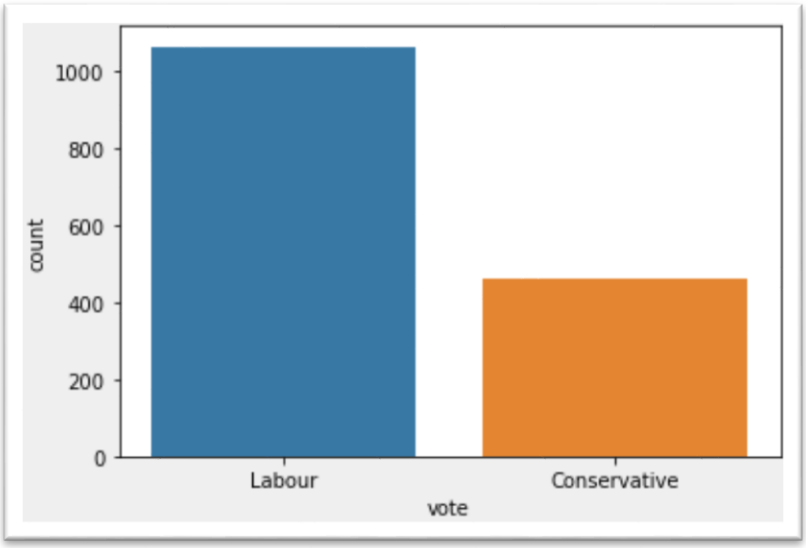


political.knowledge:

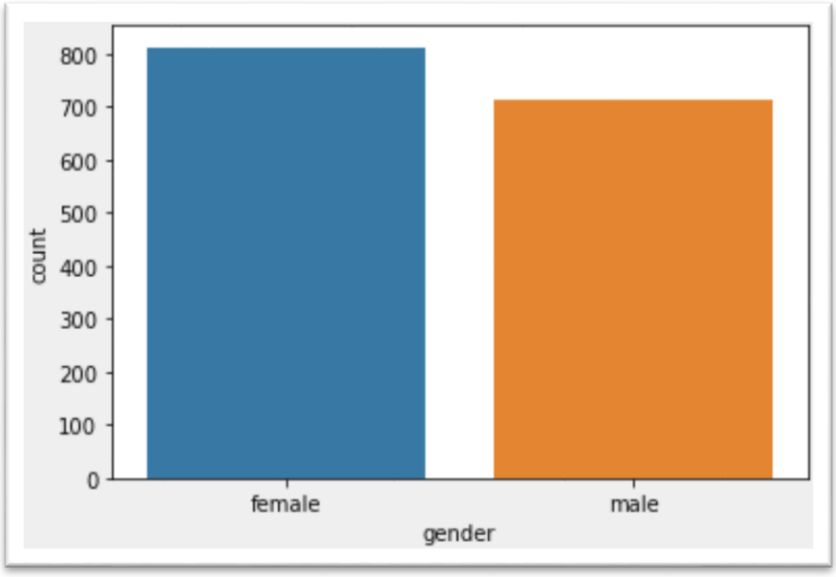
```
Range of values:  3
Minimum Age:  0
Maximum Age:  3
Mean value:  1.5422950819672132
Median value:  2.0
Standard deviation:  1.0833147486432724
Null values:  False
```



vote:



gender:



EDA:

EDA-Step-1: Checking for duplicate records in the data:

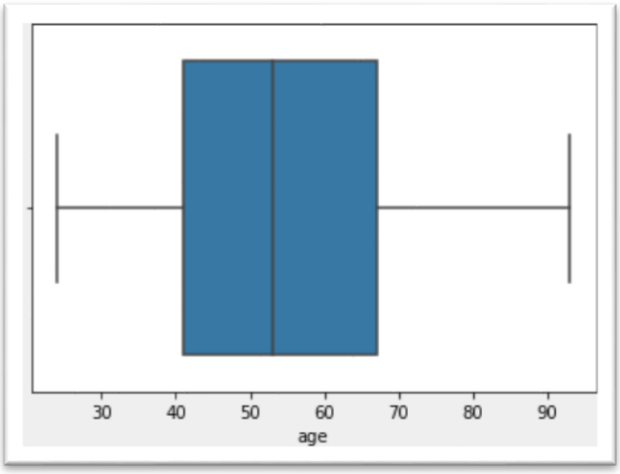
```
Total no of duplicate values = 8
```

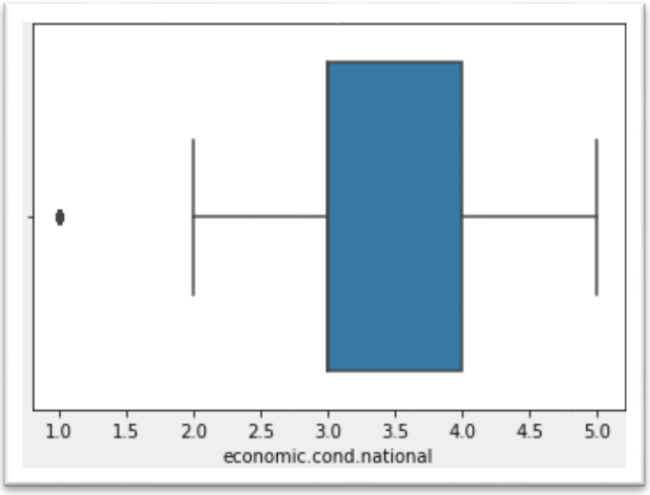
	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
67	Labour	35	4	4	5	2	3	2	male
626	Labour	39	3	4	4	2	5	2	male
870	Labour	38	2	4	2	2	4	3	male
983	Conservative	74	4	3	2	4	8	2	female
1154	Conservative	53	3	4	2	2	6	0	female
1236	Labour	36	3	3	2	2	6	2	female
1244	Labour	29	4	4	4	2	2	2	female
1438	Labour	40	4	3	4	2	2	2	male

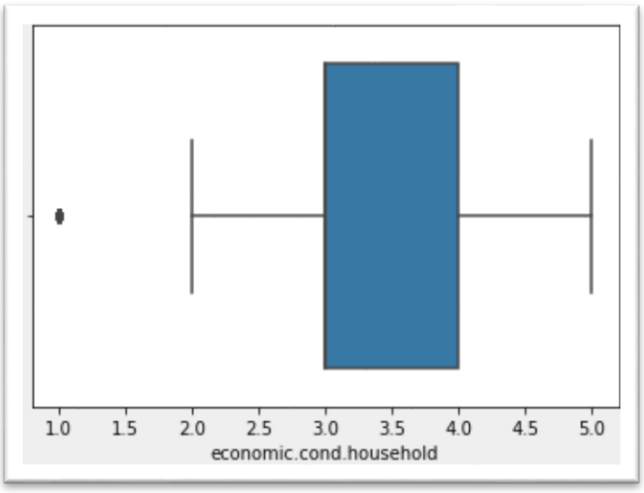
EDA-Step 2: Checking Missing value:

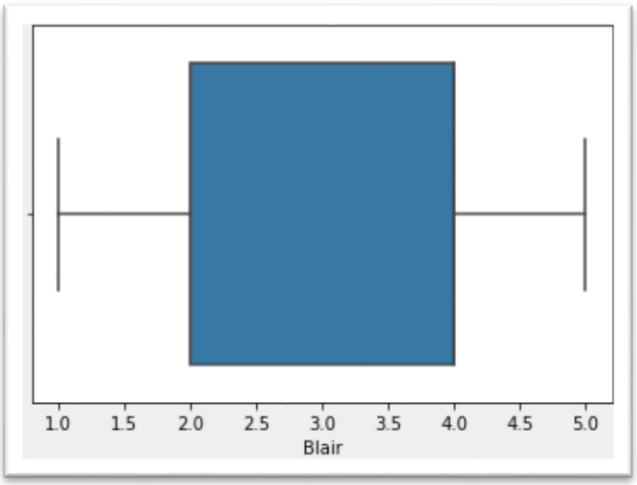
```
vote                    0
age                     0
economic.cond.national  0
economic.cond.household 0
Blair                   0
Hague                   0
Europe                  0
political.knowledge     0
gender                  0
dtype: int64
```

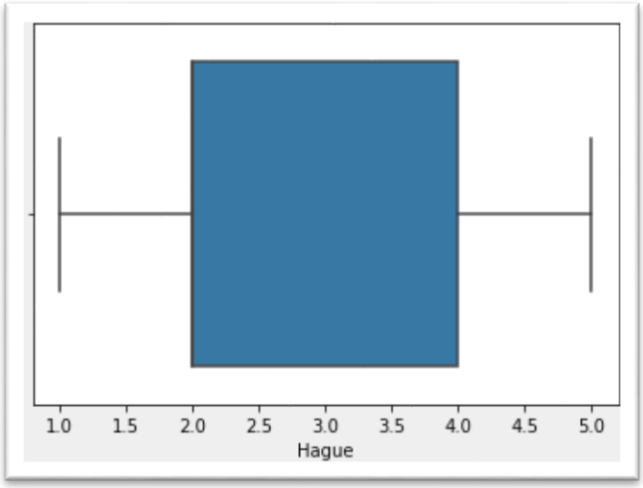
EDA-Step 3: Outlier Checks:

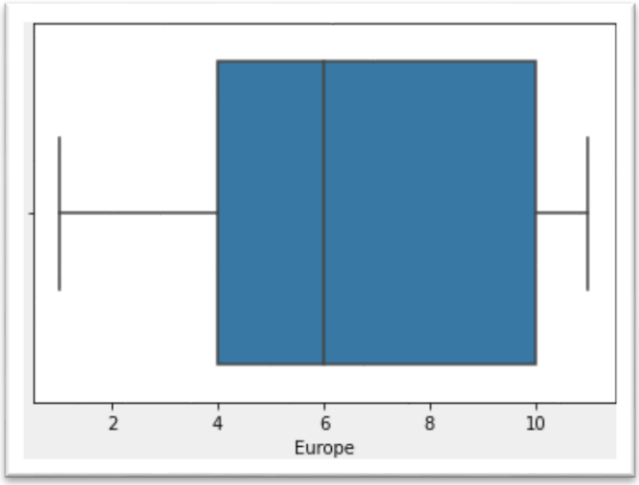


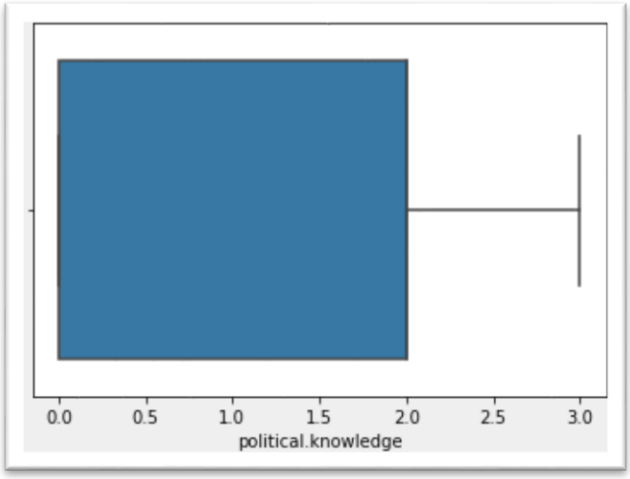




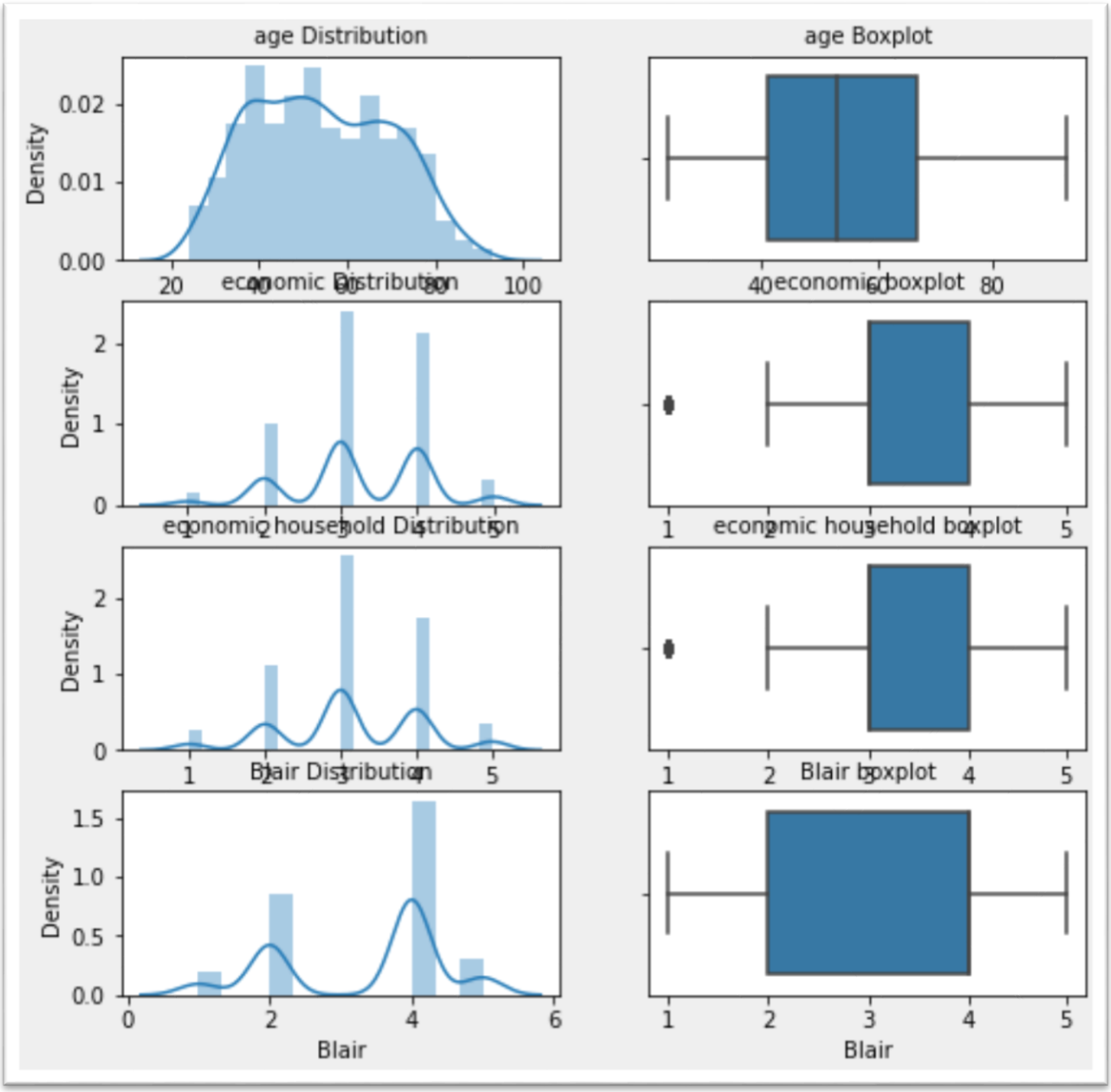


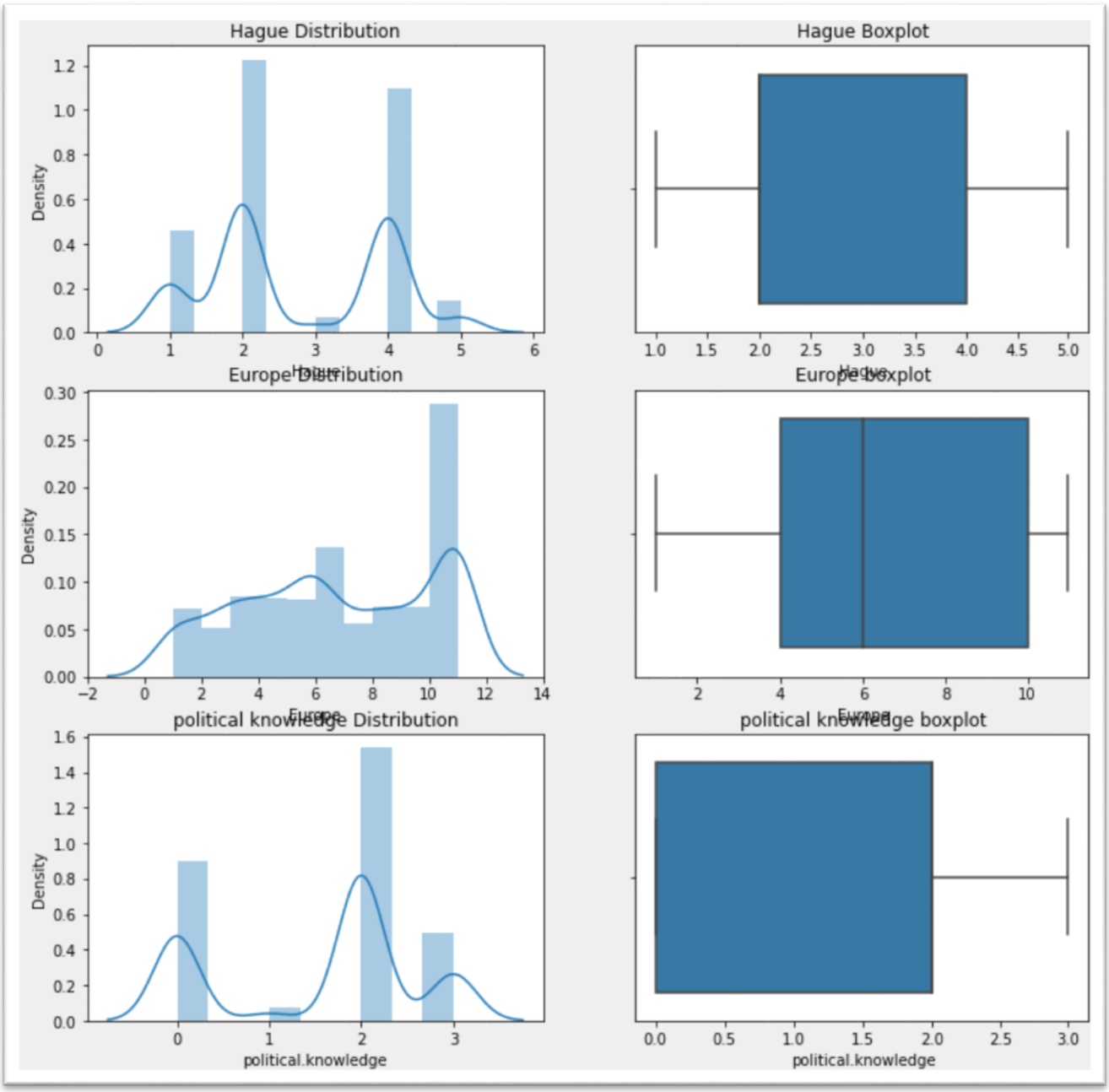






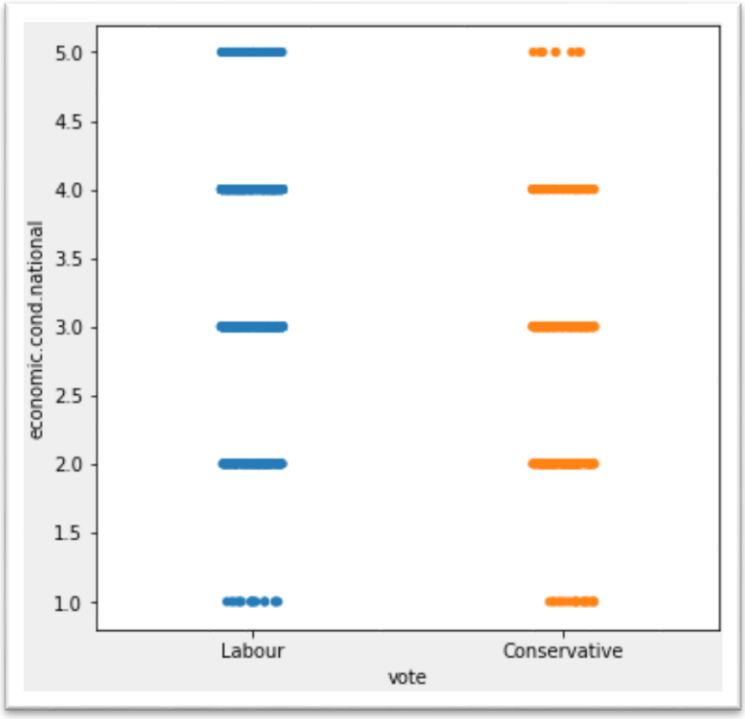
EDA-Step 4: Univariate Analysis:

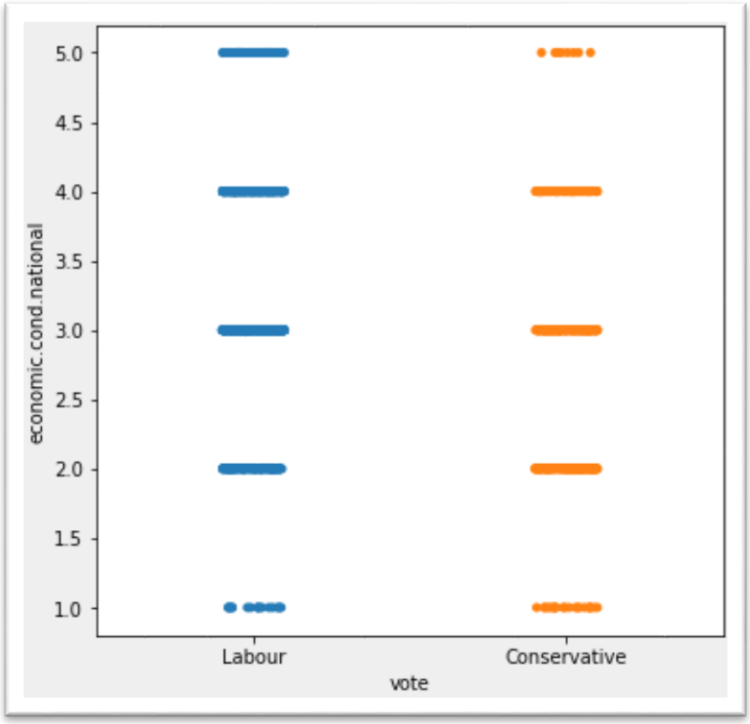


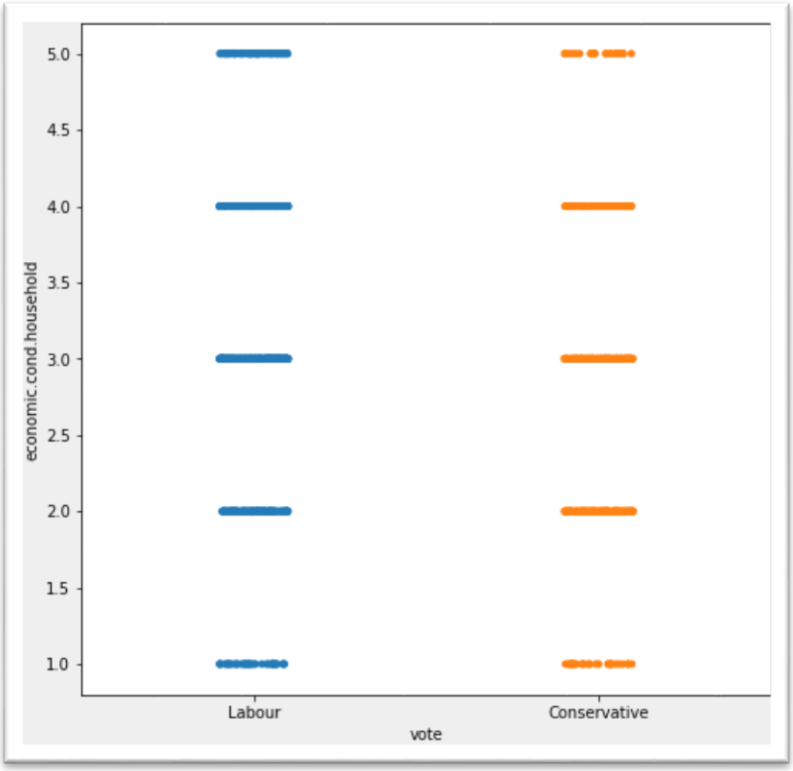


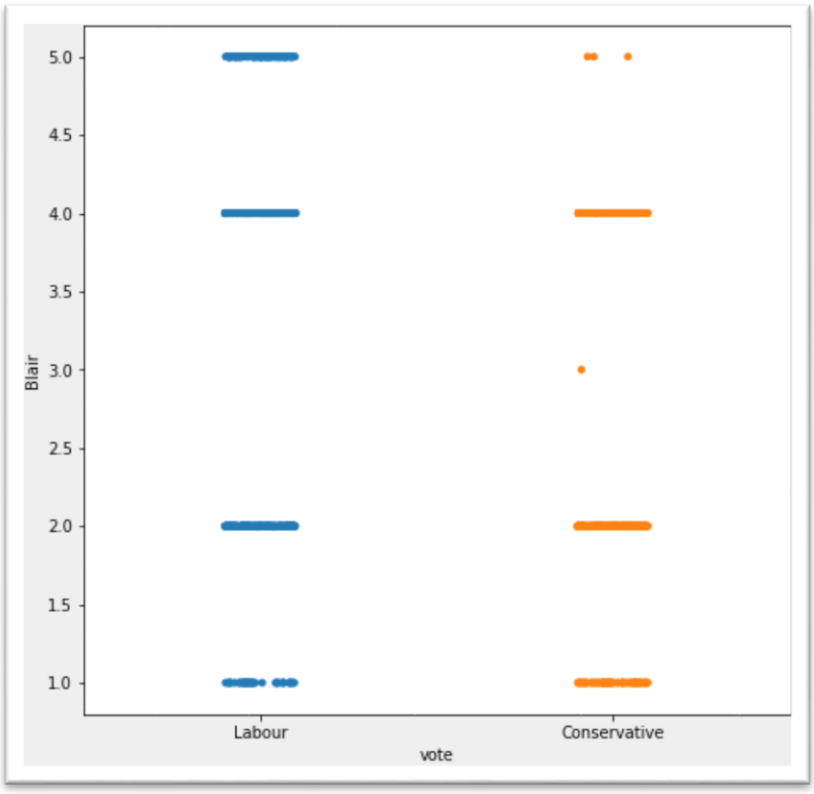
- We can see that the numerical variables are normally distributed (not perfectly normal though and are multi modal in some instances as well)
- There are outliers present in “economic_cond_national” and “economic_cond_household” variables that can be seen from box plots.
- Also here, the upper and the lower limits are not distant from each other and the outliers on the lower side only that too having value 1 where the lower limit is 1.5.
- So it is not advisable to treat the outliers in this case.
- We will move forward without treating the outliers.

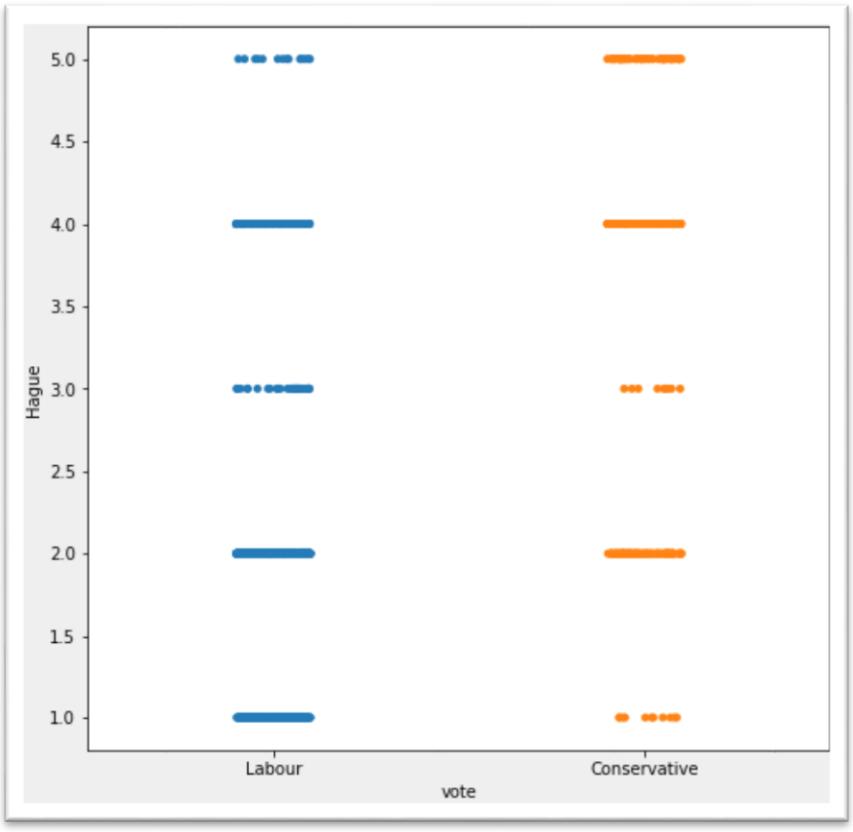
EDA-Step 5: Bivariate and Multivariate Analysis:

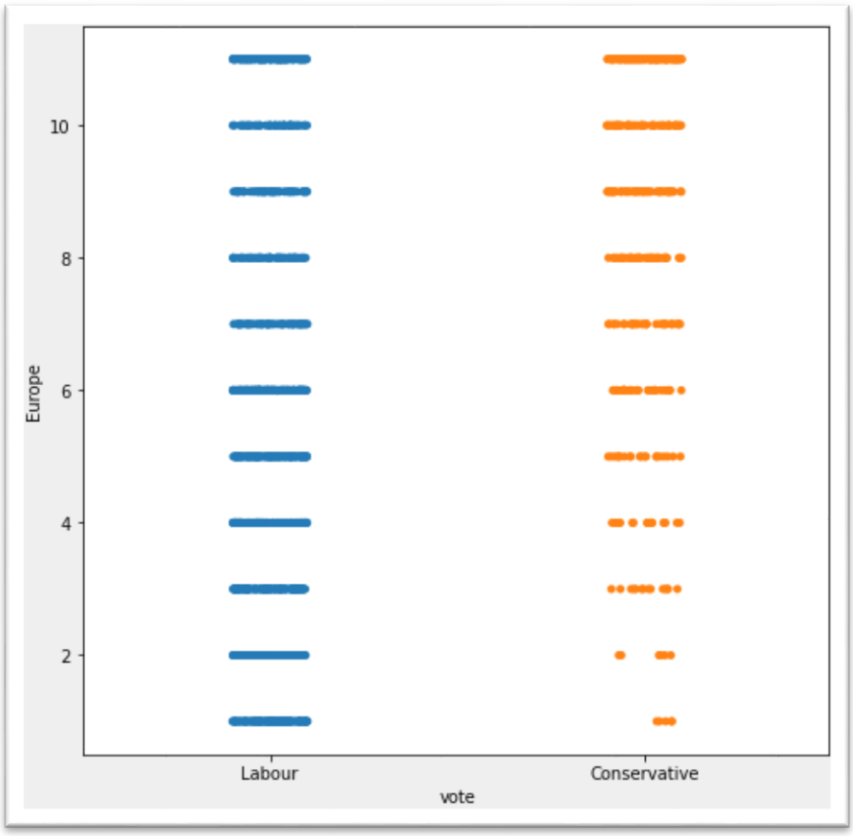


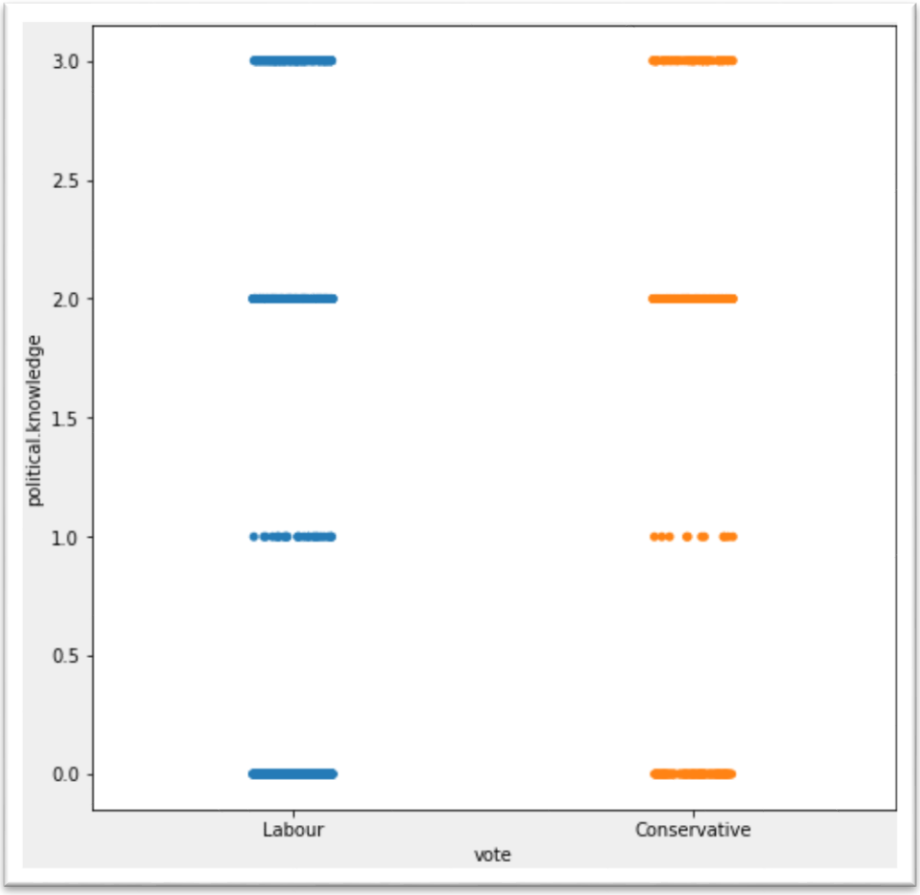


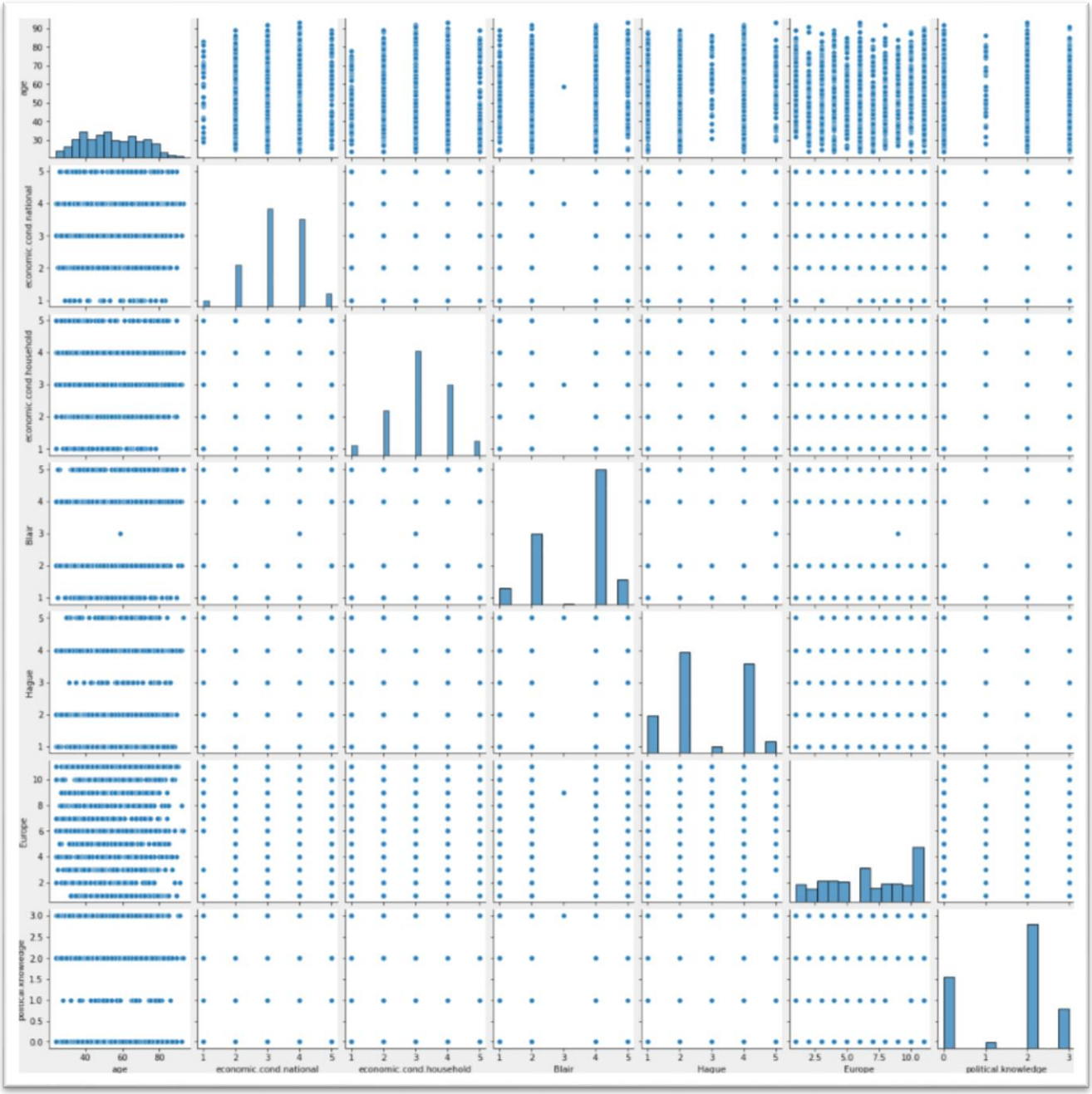




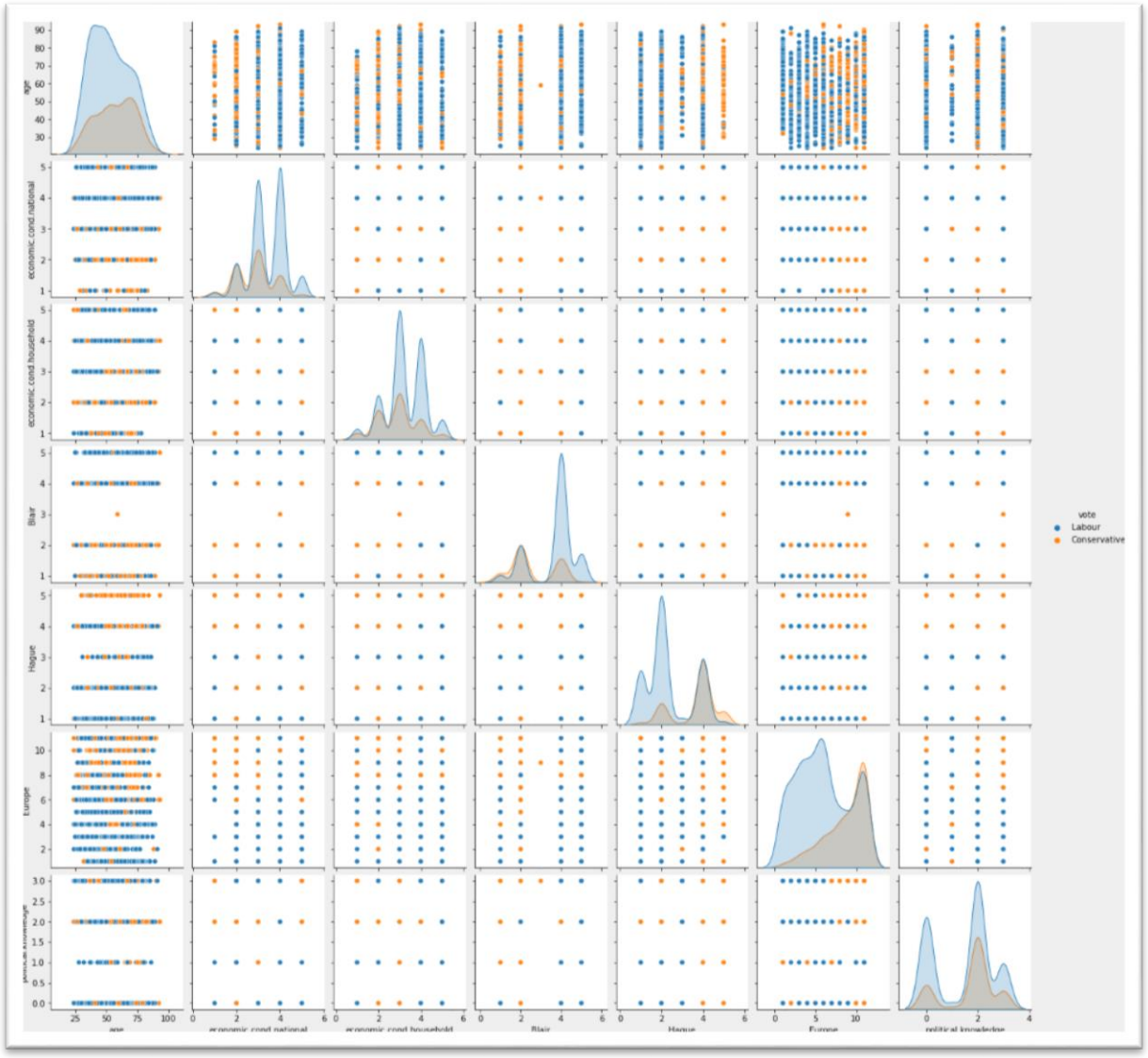


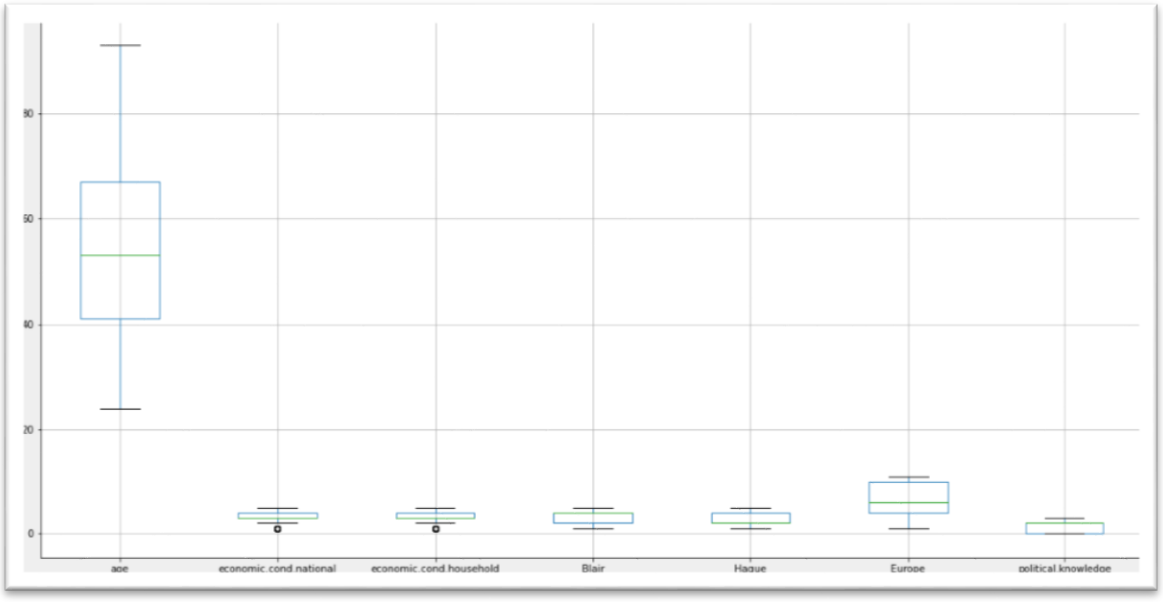




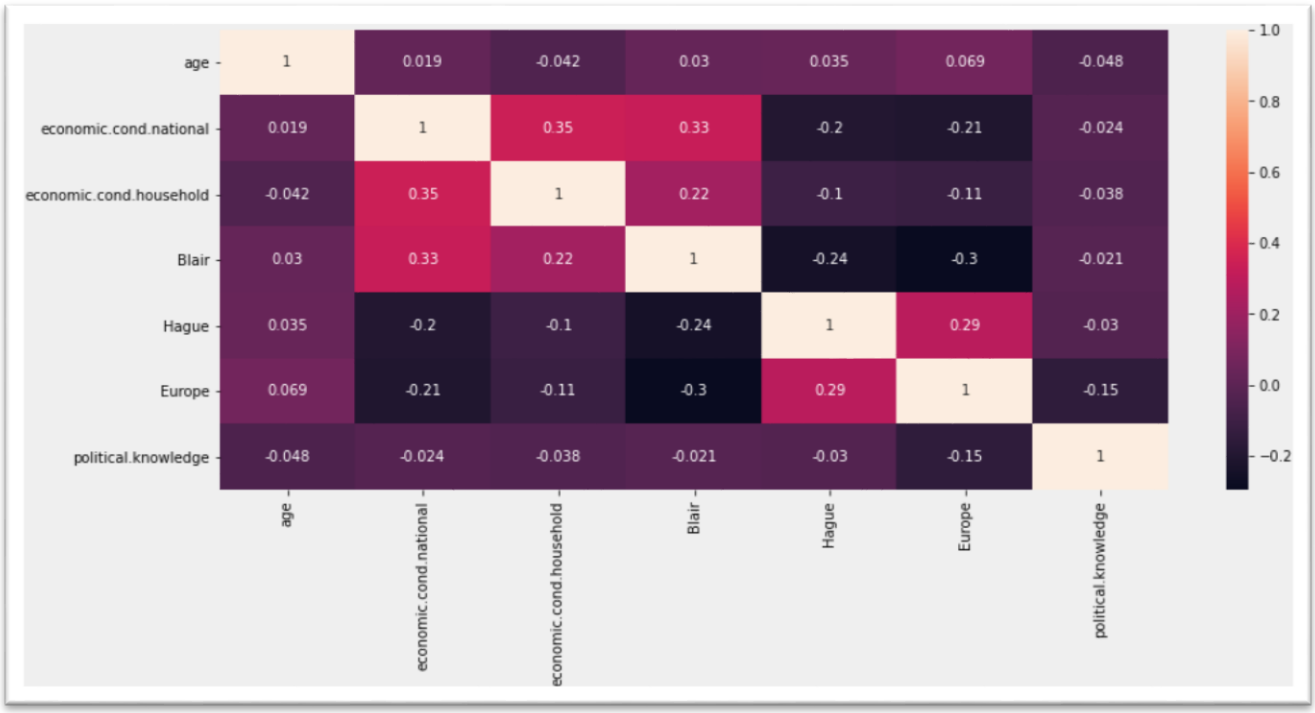


- Pairplot tells us about the interaction of each variable with every other variable present.
- As such there is no strong relation ship present between the variables.
- There is a mixture of positive and negative relationships though which is expected.
- Overall, it's a rough estimate of the interactions, clearer picture can be obtained by heat map values and also different kinds of plot.



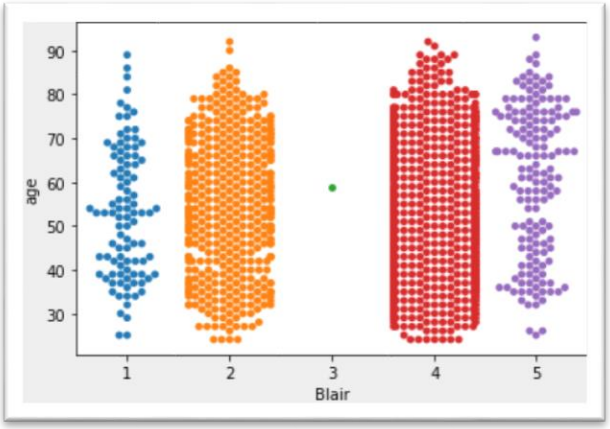


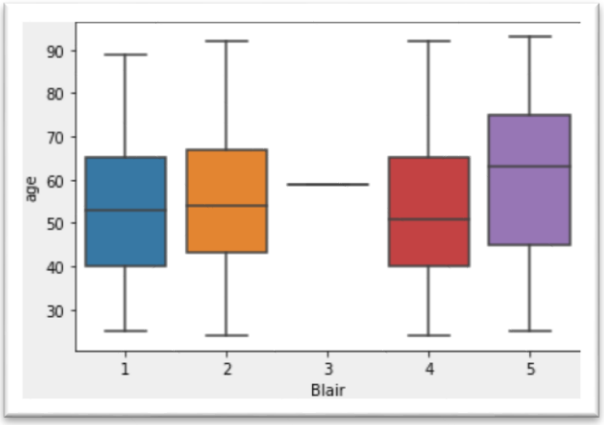
Heat Map:



- Multicollinearity is an important issue which can harm the model. Heatmap is a good way of identifying this issue. It gives us the basic idea of the relation ship the variables have with each other.
- Highest positive correlation is between “economic_cond_national” & “economic_cond_household” (35%). But the good thing is that it’s not that huge.
- Highest negative correlation is between “Blair” and “Europe” (30%) but is not huge.
- Thus, multicollinearity won’t be a issue in this data set.

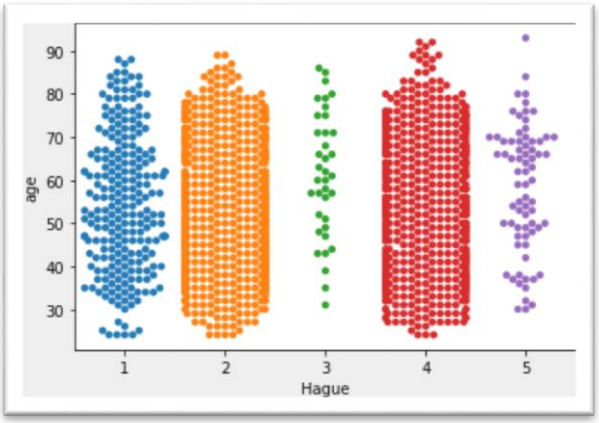
Blair Vs Age:

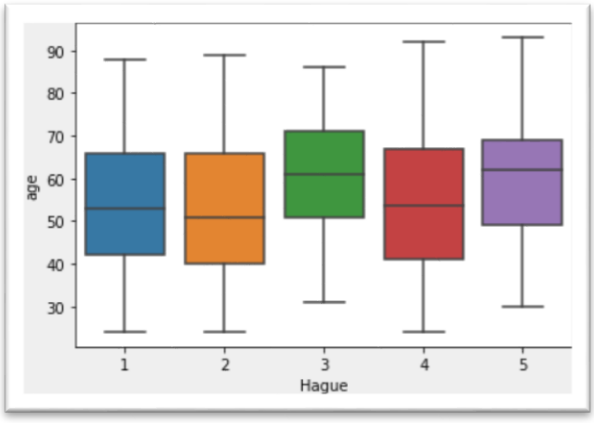




- People above the age of 45 years generally think that Blair is doing a good job.

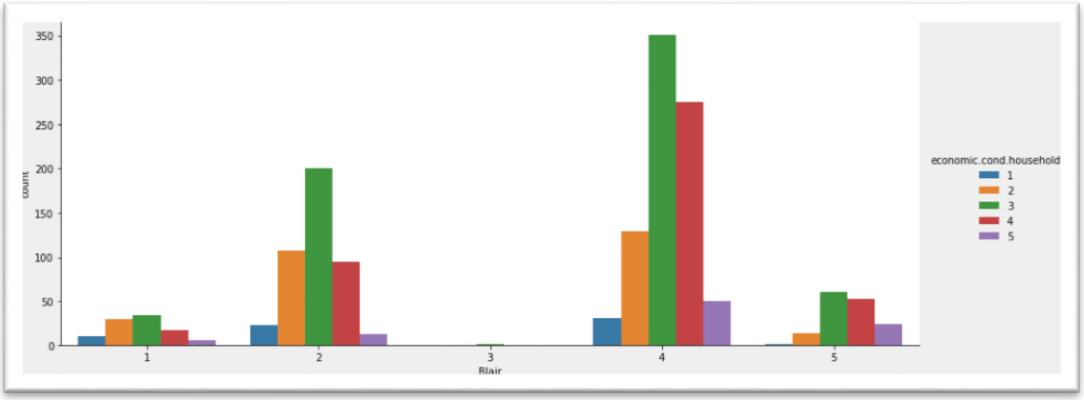
Hague Vs age:



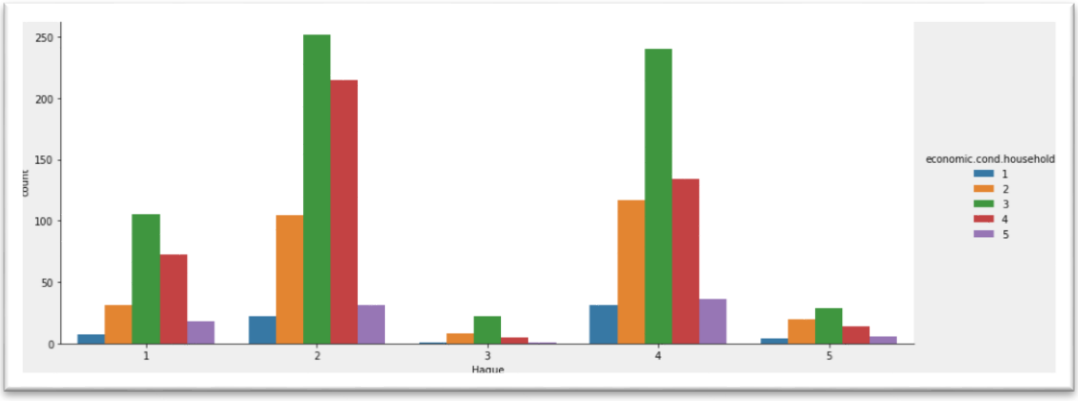


- Hague has slightly more concentration of neural points than that of the Blair for people above 50 years of age.

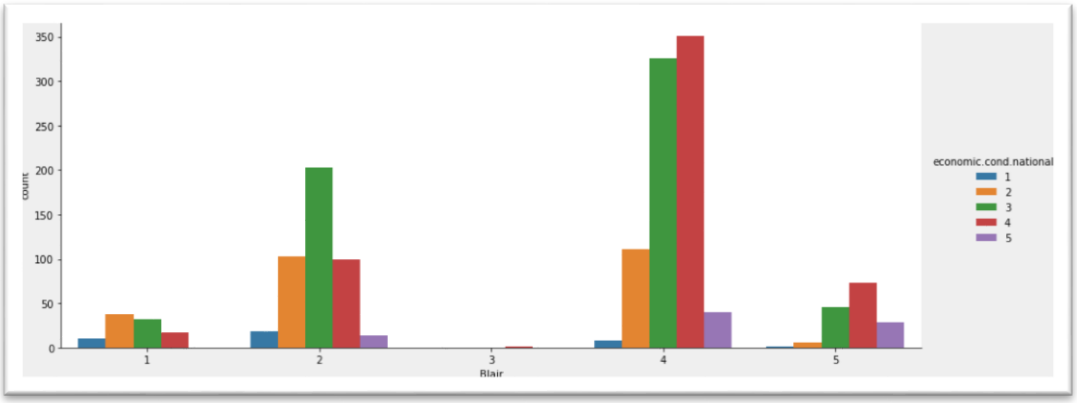
Catplot Analysis- Blair (Count) on economic.cond.household:



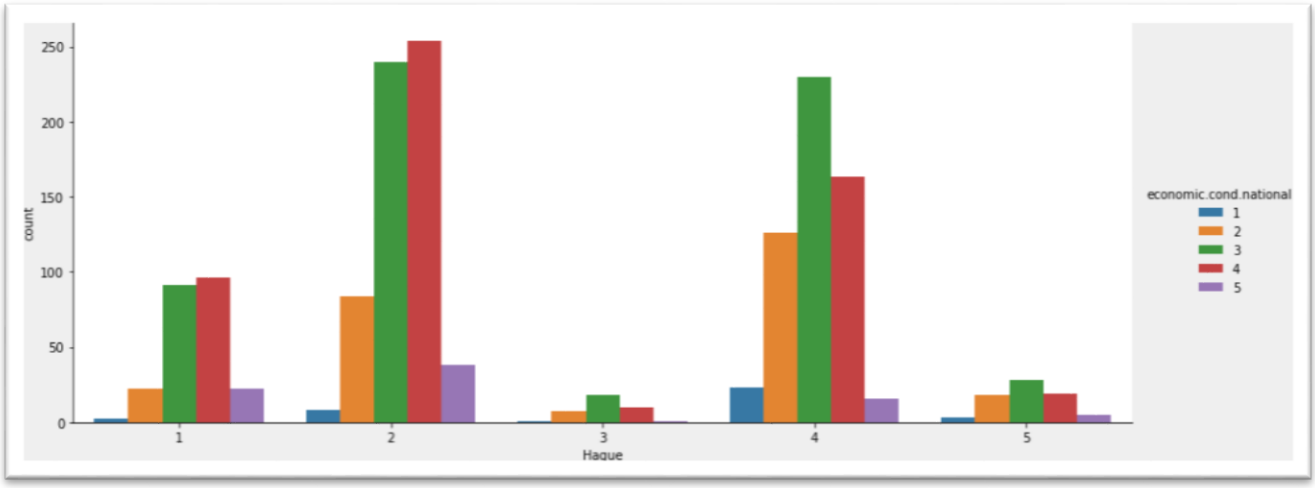
Catplot Analysis- Hague (Count) on economic.cond.household:



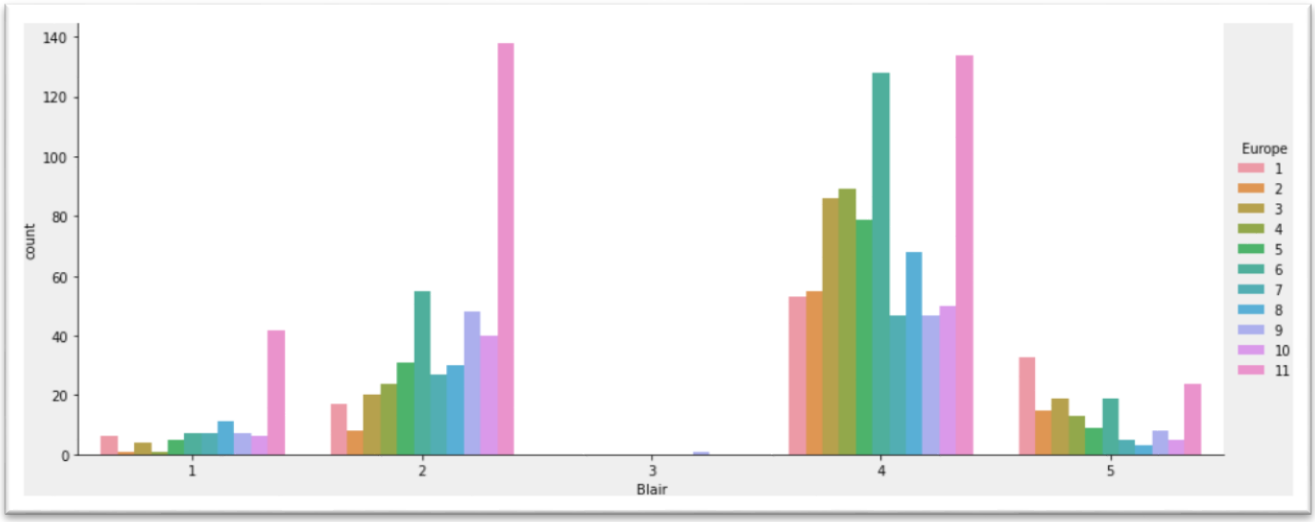
Catplot Analysis- Blair (Count) on economic.cond.national:



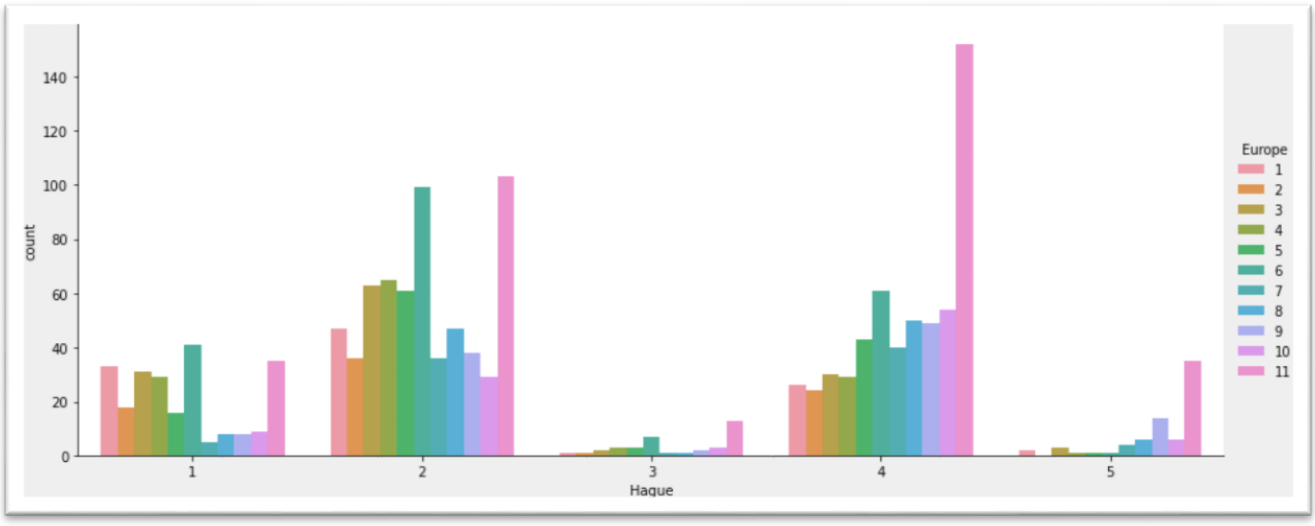
Catplot Analysis- Hague (Count) on economic.cond.national:



Catplot Analysis- Blair (Count) on Europe:

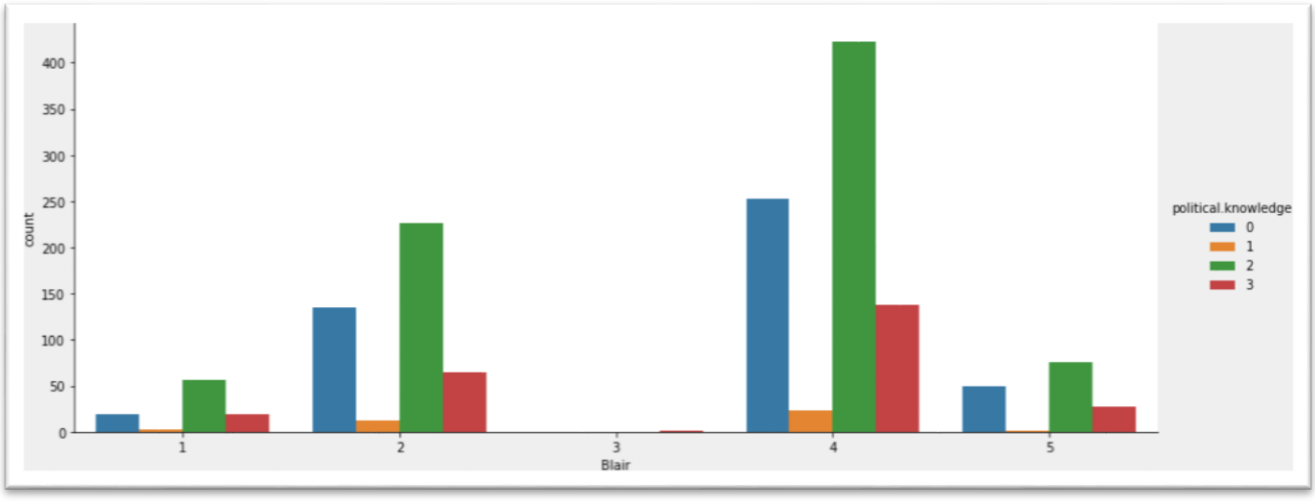


Catplot Analysis- Hague (Count) on Europe:

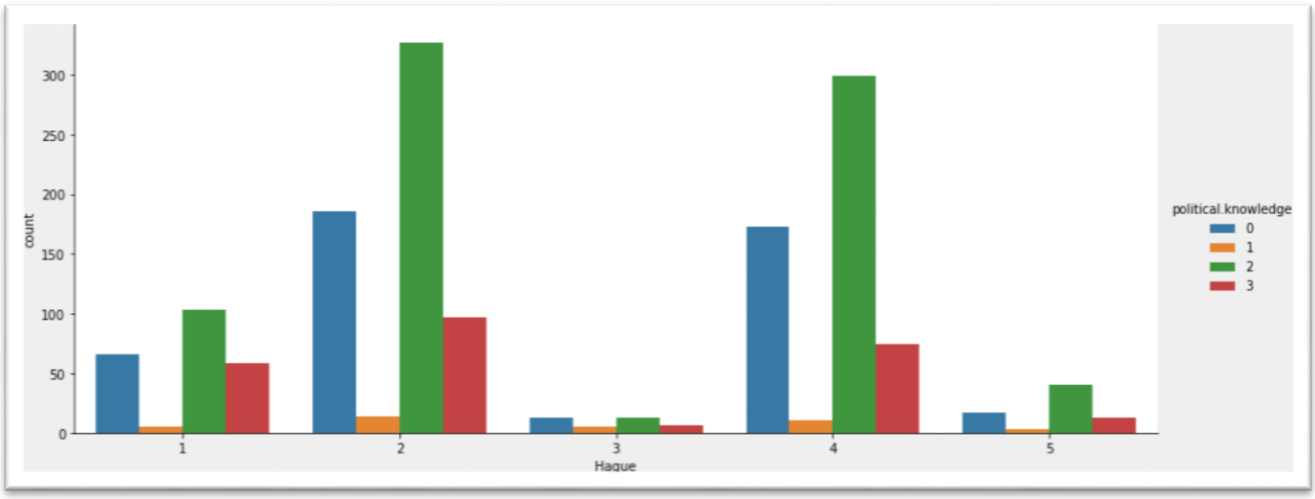


- In the whole Europe, if we look at the data, then Blair is leading.

Catplot Analysis- Blair (Count) on political.knowledge:



Catplot Analysis- Hague (Count) on political.knowledge:



- In terms of political knowledge, Blair is considered better.

	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

1.3) Encode the data (having string values) for Modelling. Is Scaling necessary here or not?(2 pts), Data Split: Split the data into train and test (70:30) (2 pts). The learner is expected to check and comment about the difference in scale of different features on the bases of appropriate measure for example std dev, variance, etc. Should justify whether there is a necessity for scaling. Object data should be converted into categorical/numerical data to fit in the models.

As many machine learning models cannot work with string values, thus we will need to encode the categorical variables and convert their data types into the integer type.;
From the info the data set, we know there are 2 categorical type variables, so we need to encode these two variables with suitable techniques.

Unique Count for Categorical variable:

```
cut
 Labour          1063
Conservative     462
Name: vote, dtype: int64


color
 female    812
male      713
Name: gender, dtype: int64
```

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	vote_Labour	gender_male
0	43	3	3	4	1	2	2	1	0
1	36	4	4	4	4	5	2	1	1
2	35	4	4	5	2	3	2	1	1
3	24	4	2	2	1	4	0	1	0
4	41	2	2	1	1	6	2	1	1

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   age                     1525 non-null   int64
 1   economic.cond.national  1525 non-null   int64
 2   economic.cond.household 1525 non-null   int64
 3   Blair                   1525 non-null   int64
 4   Hague                   1525 non-null   int64
 5   Europe                  1525 non-null   int64
 6   political.knowledge     1525 non-null   int64
 7   vote_Labour             1525 non-null   uint8
 8   gender_male             1525 non-null   uint8
dtypes: int64(7), uint8(2)
memory usage: 86.5 KB
```

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	IsLabour_or_not	IsMale_or_not
1493	34	3	1	4	2	6	2	1	0
1171	37	4	3	4	1	4	0	1	1
674	43	4	1	2	4	6	2	1	1
1179	74	4	4	4	2	9	0	1	0
28	44	3	3	4	2	1	2	1	1
1462	76	5	3	4	1	11	0	1	0
521	51	4	3	4	1	2	2	1	1
1040	59	3	3	2	4	4	3	0	0
1032	35	2	3	2	3	10	2	0	1
1208	45	4	4	4	2	6	0	1	0

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	IsMale_or_not
0	43	3	3	4	1	2	2	0
1	36	4	4	4	4	5	2	1
2	35	4	4	5	2	3	2	1
3	24	4	2	2	1	4	0	0
4	41	2	2	1	1	6	2	1

Scaling the Data set:

Scaling is done so that the data which belongs to wide variety of ranges can be brought together in similar range and thus bringing out the best performances of the model.
Generally, we perform, Feature scaling while dealing with the Gradient Descent Based algorithms such as Linear and Logistic Regression as these are very sensitive to the range in the data points. In additions, its is very useful in checking and reducing the multi collinearity in the data.
Here, we will perform scaling by the help Z scaling method to get the desired results.

After applying the Z score, below is the table:

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	IsMale_or_not
0	-0.711973	-0.279218	-0.150948	0.566716	-1.419886	-1.434426	0.422643	-0.937059
1	-1.157661	0.856268	0.924730	0.566716	1.018544	-0.524358	0.422643	1.067169
2	-1.221331	0.856268	0.924730	1.418187	-0.607076	-1.131070	0.422643	1.067169
3	-1.921698	0.856268	-1.226625	-1.136225	-1.419886	-0.827714	-1.424148	-0.937059
4	-0.839313	-1.414704	-1.226625	-1.987695	-1.419886	-0.221002	0.422643	1.067169

Splitting the data set:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y, random_state=1)
```

Data distribution (Training and Testing Data)

```
Number of rows and columns of the training set for the independent variables: (1143, 8)
Number of rows and columns of the training set for the dependent variable: (1143,)
Number of rows and columns of the test set for the independent variables: (382, 8)
Number of rows and columns of the test set for the dependent variable: (382,)
```

```
1    0.692913
0    0.307087
Name: IsLabour or not, dtype: float64
```

```
1    0.709424
0    0.290576
Name: IsLabour_or_not, dtype: float64
```

1.4) Apply Logistic Regression and LDA (Linear Discriminant Analysis) (2 pts). Interpret the inferences of both model s (2 pts). Successful implementation of each model. Logical reason behind the selection of different values for the parameters involved in each model. Calculate Train and Test Accuracies for each model. Comment on the validness of models (over fitting or under fitting)

Discriminant Analysis:

Importing the LDA:

```
from sklearn.discriminant analysis import LinearDiscriminantAnalysis
```

Splitting the Data set:

```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(X,Y, random_state=1)
```

Fitting the Data Set:

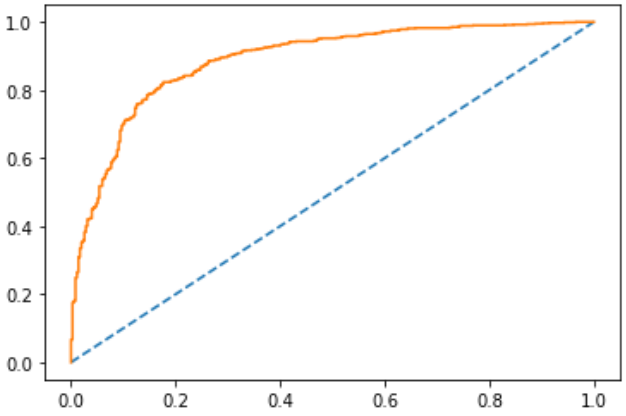
```
LDA_model=LinearDiscriminantAnalysis()
LDA_model.fit(X_train,Y_train)
```

```
LinearDiscriminantAnalysis()
```

For Training Data Set:

```
0.8372703412073491
[[246 105]
 [ 81 711]]
              precision    recall  f1-score   support

           0       0.75      0.70      0.73       351
           1       0.87      0.90      0.88       792

    accuracy                           0.84      1143
   macro avg       0.81      0.80      0.80      1143
weighted avg       0.83      0.84      0.84      1143
```

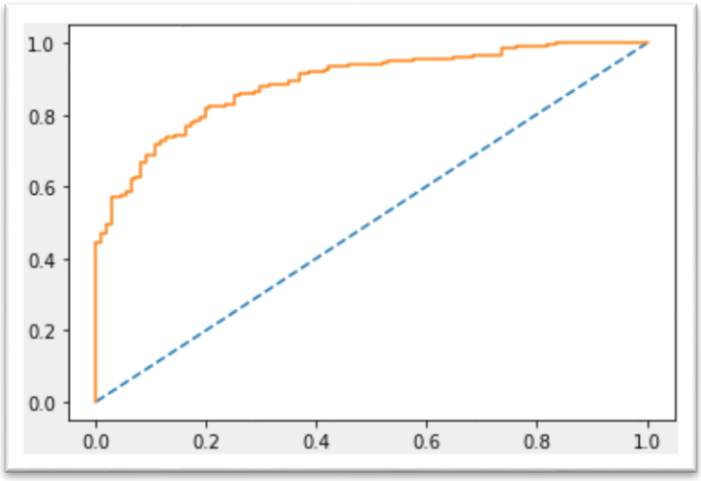
AUC ROC curve for LDA Train:



For Testing Data Set:

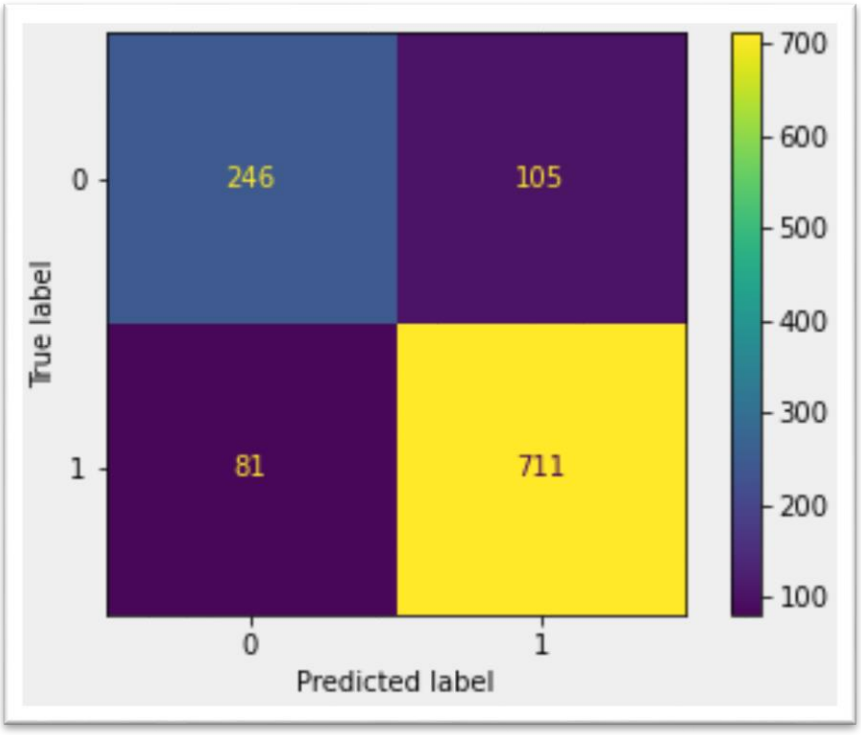
```
0.824607329842932
[[ 76  35]
 [ 32 239]]
              precision    recall  f1-score   support

           0       0.70      0.68      0.69       111
           1       0.87      0.88      0.88       271

    accuracy                           0.82       382
   macro avg       0.79      0.78      0.79       382
weighted avg       0.82      0.82      0.82       382
```

AUC ROC curve for LDA Test:



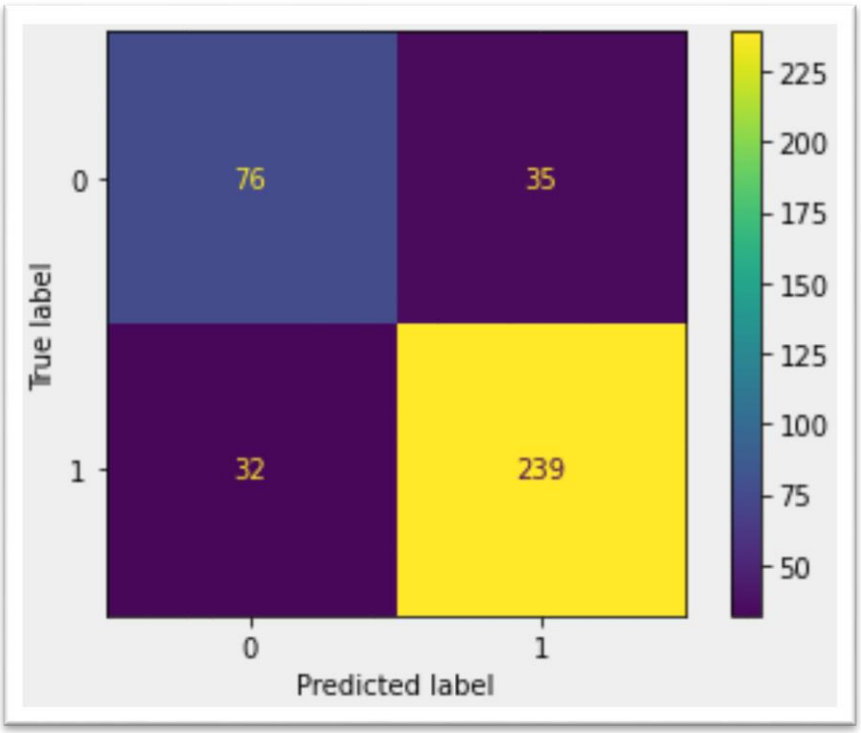
Confusion Matrix for the training data:

```
array([[242, 109],
       [ 73, 719]], dtype=int64)
```



Confusion Matrix for test data:

```
array([[ 74,  37],
       [ 31, 240]], dtype=int64)
```



Logistic Regression:

Importing the Logistic Regression:

```
from sklearn.linear_model import LogisticRegression
```

Fitting the Data set

```
Logistic_model = LogisticRegression(solver='newton-cg',max_iter=10000,penalty='none',verbose=True,n_jobs=2)
Logistic_model.fit(X_train, Y_train)
```

```
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   1 out of   1 | elapsed:    0.9s finished
```

```
LogisticRegression(max_iter=10000, n_jobs=2, penalty='none', solver='newton-cg',
                   verbose=True)
```

For Training Data set:

```
0.8407699037620298
[[242 109]
 [ 73 719]]
              precision    recall  f1-score   support

           0       0.77      0.69      0.73       351
           1       0.87      0.91      0.89       792

    accuracy                           0.84      1143
   macro avg       0.82      0.80      0.81      1143
weighted avg       0.84      0.84      0.84      1143
```

Training Probabilities:

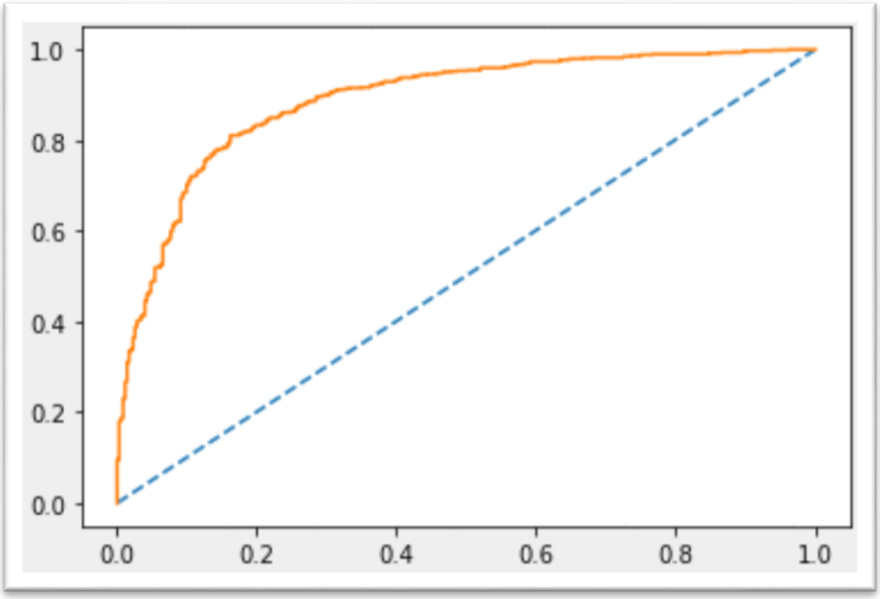
	0	1
0	0.737788	0.262212
1	0.022668	0.977332
2	0.957092	0.042908
3	0.026836	0.973164
4	0.049037	0.950963

Model Score:

```
Logistic_model.score(X_train,Y_train)
```

```
0.8407699037620298
```

AUC ROC curve for Logistic Regression Train:



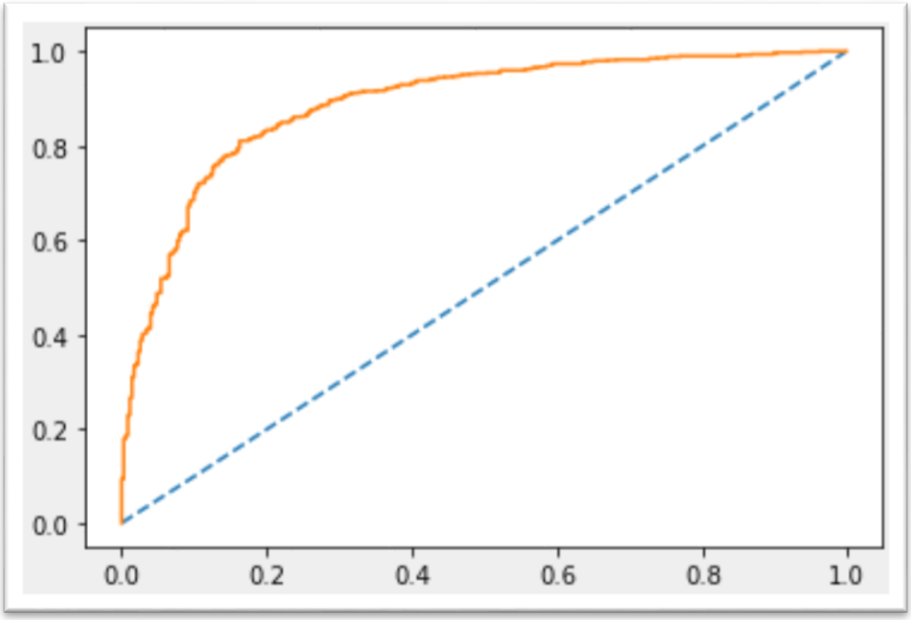
For Training Data set:

```
0.8219895287958116
[[ 74  37]
 [ 31 240]]
              precision    recall  f1-score   support

           0       0.70      0.67      0.69       111
           1       0.87      0.89      0.88       271

    accuracy                           0.82       382
   macro avg       0.79      0.78      0.78       382
weighted avg       0.82      0.82      0.82       382
```

Testing Probabilities:

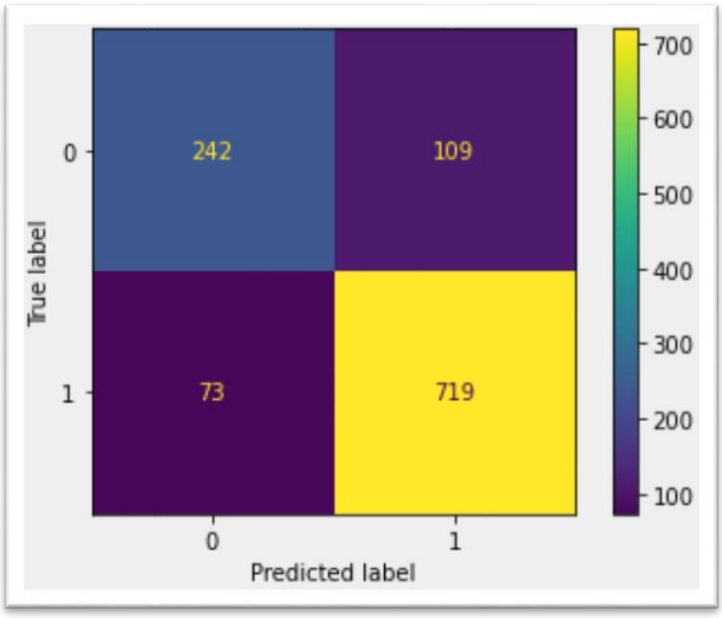
	0	1
0	0.927149	0.072851
1	0.699505	0.300495
2	0.326127	0.673873
3	0.487992	0.512008
4	0.171596	0.828404

AUC ROC curve for Logistic Regression Test:



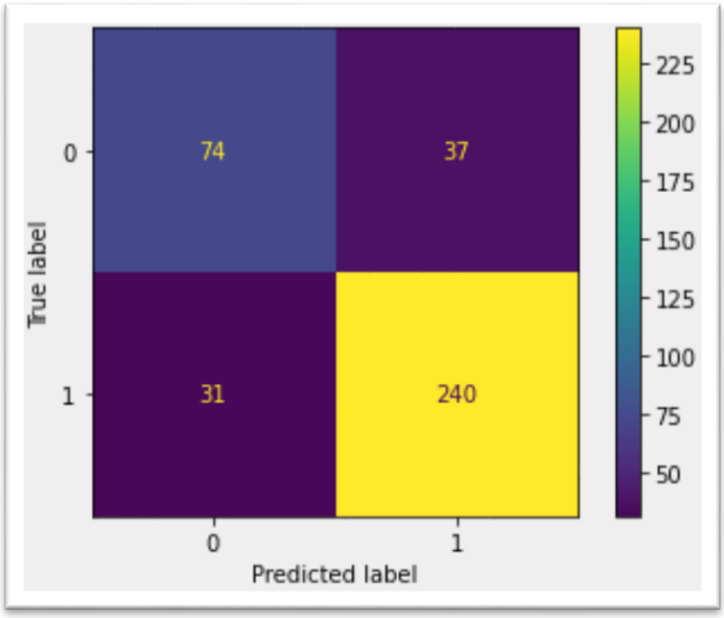
Confusion Matrix for the training data:

```
array([[242, 109],
       [ 73, 719]], dtype=int64)
```



Confusion Matrix for test data:

```
array([[ 74,  37],
       [ 31, 240]], dtype=int64)
```



1.5) Apply KNN Model and Naïve Bayes Model (2pts). Interpret the inferences of each model (2 pts). Successful implementation of each model. Logical reason behind the selection of different values for the parameters involved in each model. Calculate Train and Test Accuracies for each model. Comment on the validness of models (over fitting or under fitting)

Naive Bayes:

```
X=df.drop('IsLabour_or_not',axis=1)
Y=df['IsLabour_or_not']
```

Splitting the Data Set:

```
X_train,X_test, Y_train, Y_test=train_test_split(X,Y,train_size=0.70, random_state=1
```

Importing the Naïve Bayes:

```
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics
```

Fitting the Data Set

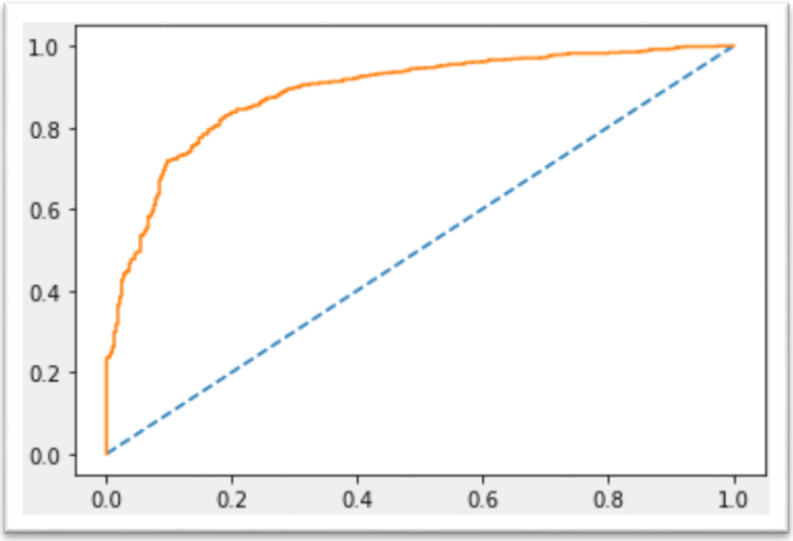
```
NB_model=GaussianNB()
NB_model.fit(X_train, Y_train)
```

```
GaussianNB()
```

For Training Data Set:

```
0.8331771321462043
[[240  92]
 [ 86 649]]
              precision    recall  f1-score   support

           0       0.74      0.72      0.73       332
           1       0.88      0.88      0.88       735

    accuracy                           0.83      1067
   macro avg       0.81      0.80      0.80      1067
weighted avg       0.83      0.83      0.83      1067
```

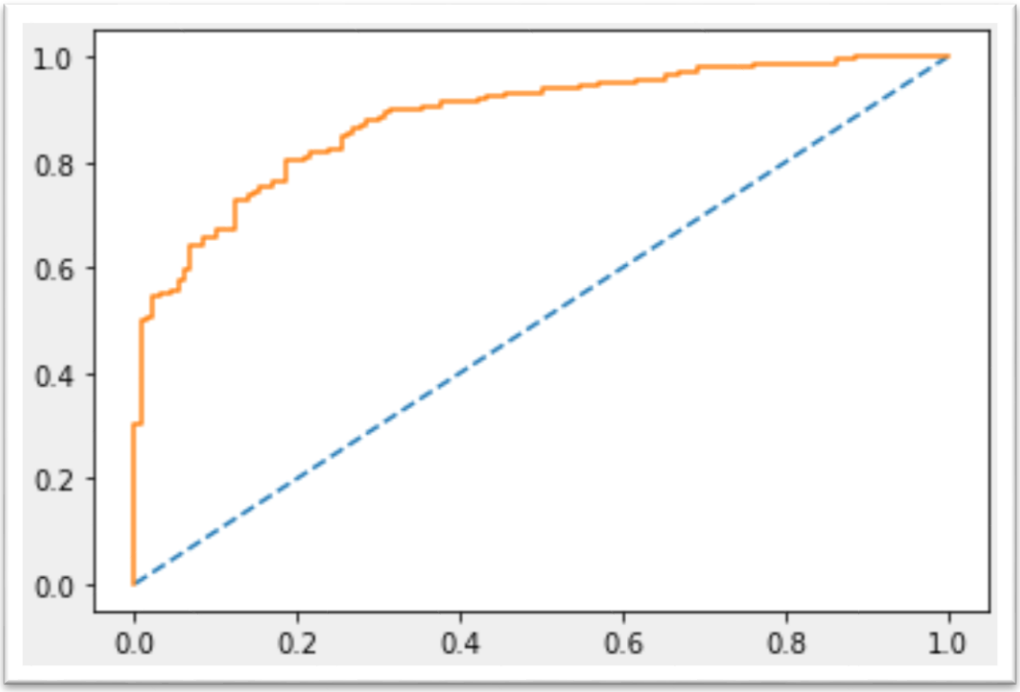
AUC Curves:



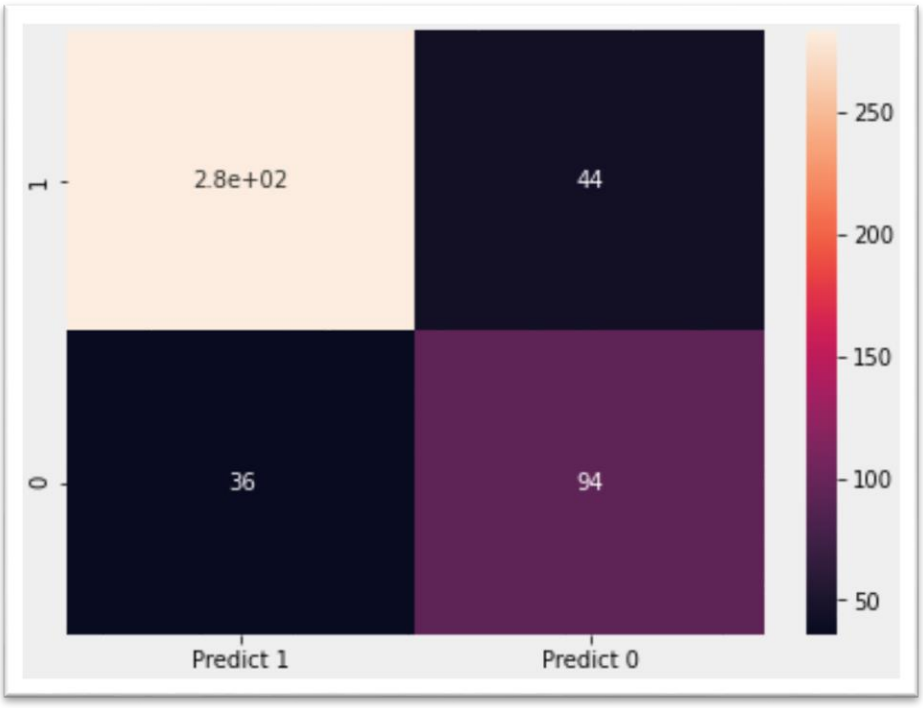
For Testing Data Set:

```
0.8253275109170306
[[ 94  36]
 [ 44 284]]
              precision    recall  f1-score   support

           0       0.68      0.72      0.70       130
           1       0.89      0.87      0.88       328

    accuracy                           0.83       458
   macro avg       0.78      0.79      0.79       458
weighted avg       0.83      0.83      0.83       458
```

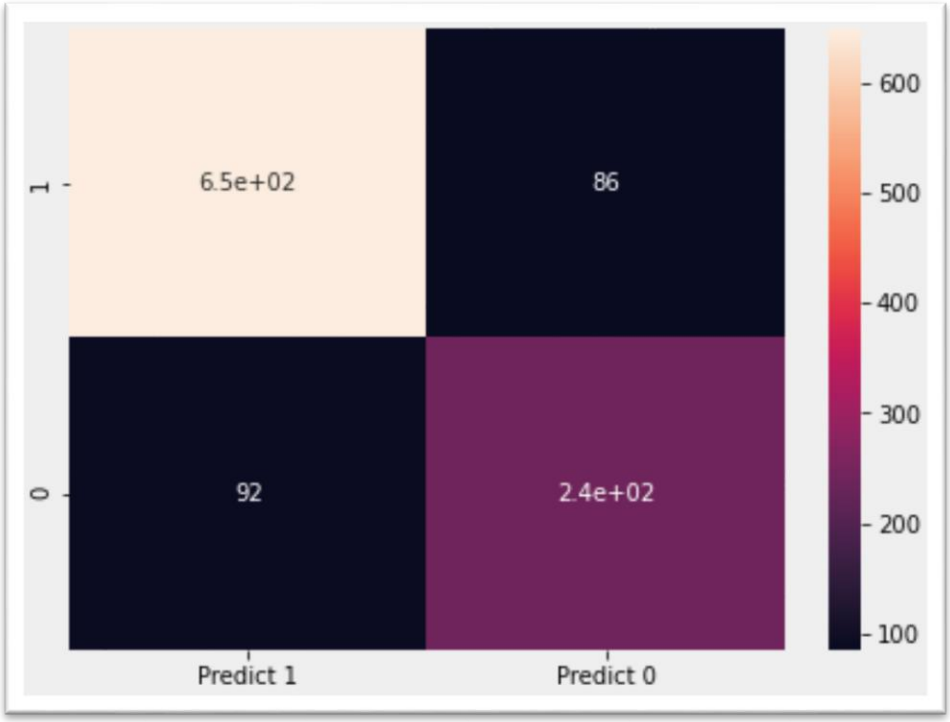
AUC Curves:



Confusion Matrix for the testing data:



Confusion Matrix for the training data:



KNN:

```
x=df.drop('IsLabour_or_not',axis=1)
y=df['IsLabour_or_not']
```

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	IsMale_or_not
0	43	3	3	4	1	2	2	0
1	36	4	4	4	4	5	2	1
2	35	4	4	5	2	3	2	1
3	24	4	2	2	1	4	0	0
4	41	2	2	1	1	6	2	1

After applying the Z score. We get the below mentioned results:

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	IsMale_or_not
0	-0.711973	-0.279218	-0.150948	0.566716	-1.419886	-1.434426	0.422643	-0.937059
1	-1.157661	0.856268	0.924730	0.566716	1.018544	-0.524358	0.422643	1.067169
2	-1.221331	0.856268	0.924730	1.418187	-0.607076	-1.131070	0.422643	1.067169
3	-1.921698	0.856268	-1.226625	-1.136225	-1.419886	-0.827714	-1.424148	-0.937059
4	-0.839313	-1.414704	-1.226625	-1.987695	-1.419886	-0.221002	0.422643	1.067169
5	-0.457295	-0.279218	0.924730	0.566716	1.018544	-0.827714	0.422643	1.067169
6	0.179402	-1.414704	-1.226625	0.566716	1.018544	1.295778	0.422643	1.067169
7	1.452797	-0.279218	0.924730	0.566716	-1.419886	-1.737782	-1.424148	1.067169
8	-0.966652	-0.279218	-0.150948	0.566716	1.018544	1.295778	-1.424148	-0.937059
9	1.007109	-0.279218	-1.226625	1.418187	-1.419886	1.295778	0.422643	1.067169

Splitting the Data:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y, random_state=1)
```

Fitting the Data:

```
from sklearn.neighbors import KNeighborsClassifier

KNN_model=KNeighborsClassifier()
KNN_model.fit(x_train,y_train)
```

```
KNeighborsClassifier()
```

```
y_train_predict=KNN_model.predict(x_train)
KNN_model_score=KNN_model.score(x_train,y_train)
```

Model Score:

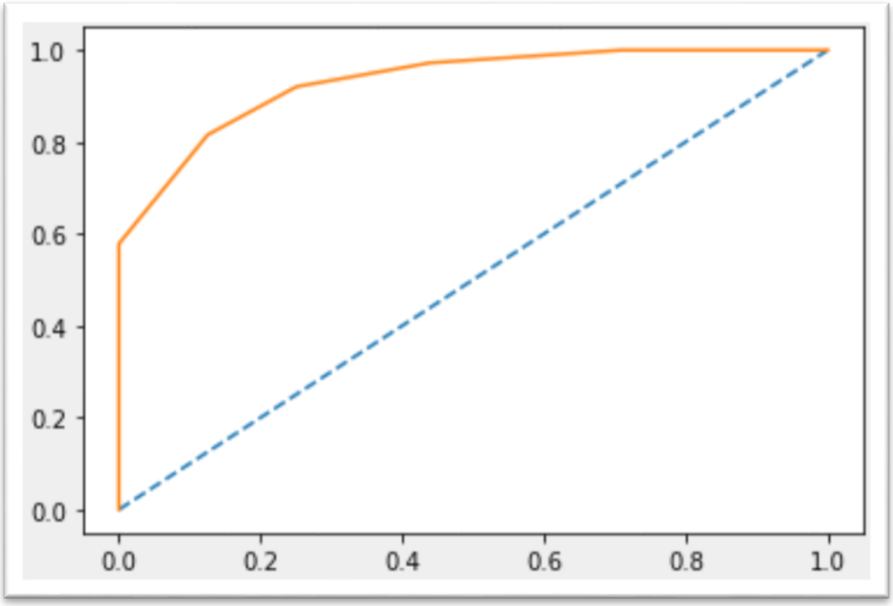
```
print(KNN_model_score)
```

```
0.8678915135608049
```

For Training Data Set:

```
[[263  88]
 [ 63 729]]
              precision    recall  f1-score   support

           0       0.81      0.75      0.78       351
           1       0.89      0.92      0.91       792

    accuracy                           0.87      1143
   macro avg       0.85      0.83      0.84      1143
weighted avg       0.87      0.87      0.87      1143
```

AUC ROC Curve KNN Train:



Model Score:

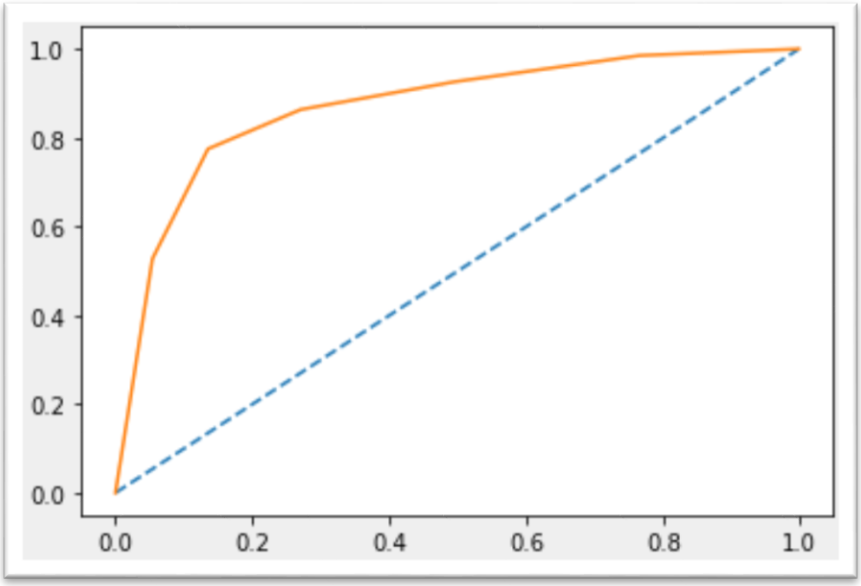
```
y_test_predict=KNN_model.predict(x_test)

KNN_model_score=KNN_model.score(x_test, y_test)

print(KNN_model_score)
```

```
0.824607329842932
```

For Testing Data Set:

```
[[ 81  30]
 [ 37 234]]
              precision    recall  f1-score   support

           0       0.69      0.73      0.71       111
           1       0.89      0.86      0.87       271

    accuracy                           0.82       382
   macro avg       0.79      0.80      0.79       382
weighted avg       0.83      0.82      0.83       382
```

AUC ROC Curve KNN Test:



For K=7 (7 Neighbors)

For Training set:

```
0.8530183727034121
[[253  98]
 [ 70 722]]
              precision    recall  f1-score   support

           0       0.78      0.72      0.75       351
           1       0.88      0.91      0.90       792

    accuracy                           0.85      1143
   macro avg       0.83      0.82      0.82      1143
weighted avg       0.85      0.85      0.85      1143
```

For Testing set:

```
0.8350785340314136
[[ 84  27]
 [ 36 235]]
              precision    recall  f1-score   support

           0       0.70      0.76      0.73       111
           1       0.90      0.87      0.88       271

    accuracy                           0.84       382
   macro avg       0.80      0.81      0.80       382
weighted avg       0.84      0.84      0.84       382
```

For K=5 (5 Neighbors)

For Training set:

```
0.8678915135608049
[[263  88]
 [ 63 729]]
              precision    recall  f1-score   support

           0       0.81      0.75      0.78       351
           1       0.89      0.92      0.91       792

    accuracy                           0.87      1143
   macro avg       0.85      0.83      0.84      1143
weighted avg       0.87      0.87      0.87      1143
```

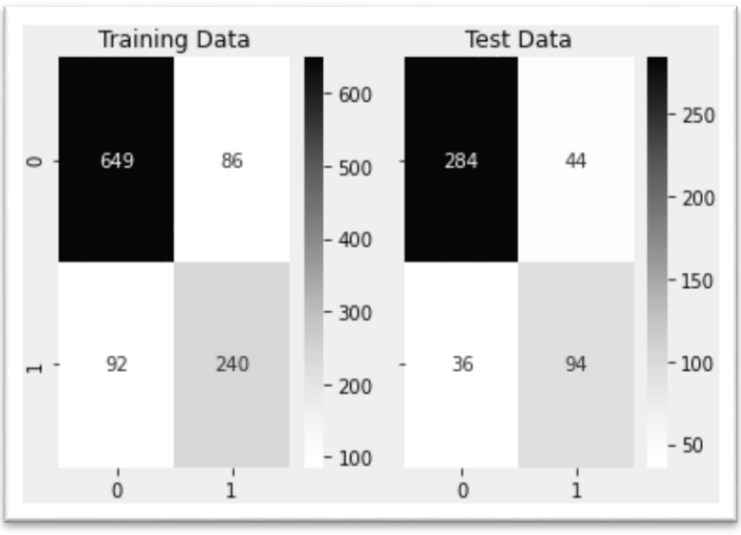
For Testing set:

```
0.824607329842932
[[ 81  30]
 [ 37 234]]
              precision    recall  f1-score   support

           0       0.69      0.73      0.71       111
           1       0.89      0.86      0.87       271

    accuracy                           0.82       382
   macro avg       0.79      0.80      0.79       382
weighted avg       0.83      0.82      0.83       382
```

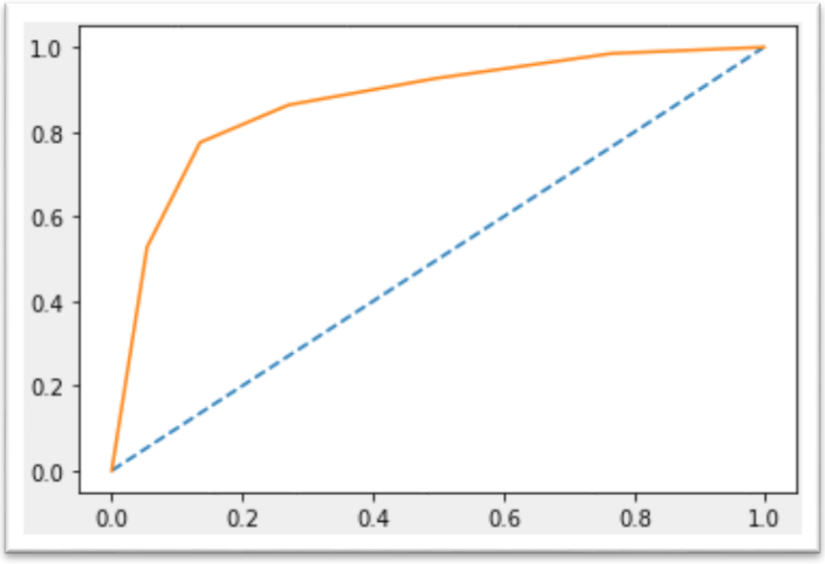
Accuracy Scores:

```
[0.23298429319371727,
 0.19633507853403143,
 0.17539267015706805,
 0.16492146596858637,
 0.17801047120418845,
 0.17277486910994766,
 0.17539267015706805,
 0.18586387434554974,
 0.17801047120418845,
 0.17277486910994766]
```

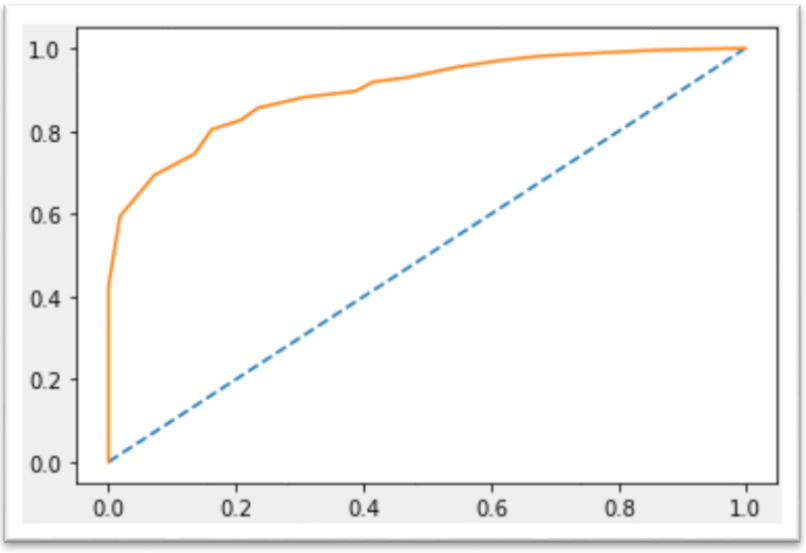
Confusion Matrix:



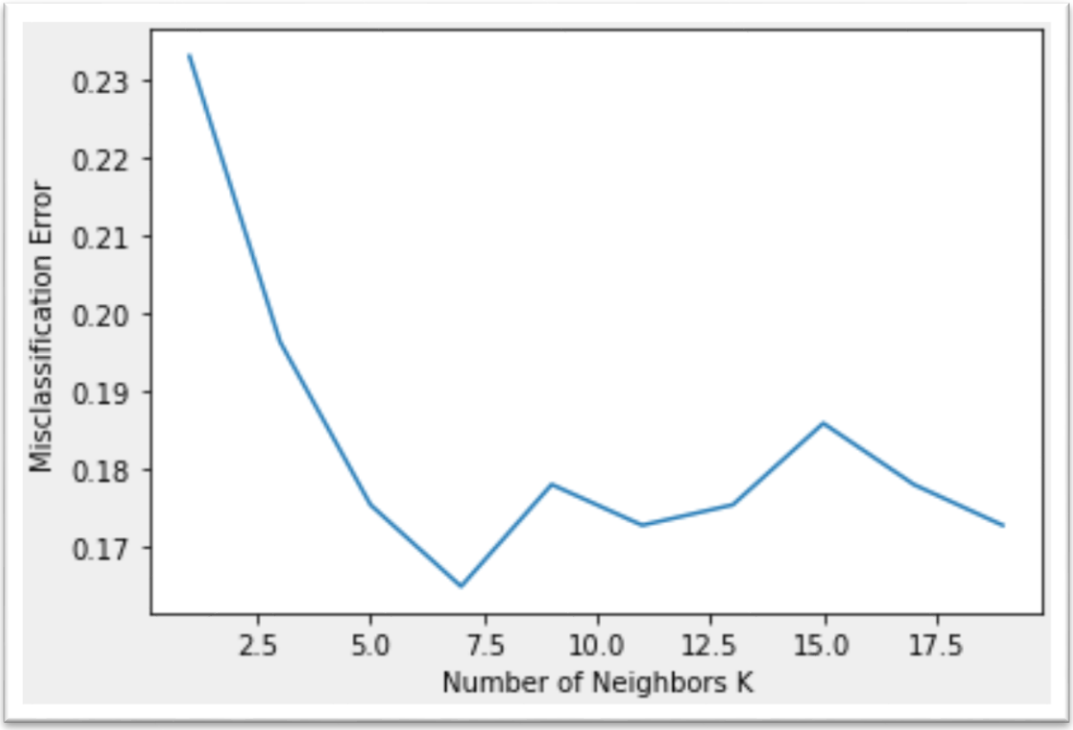
AUC ROC curve after n classifier for train data set:



AUC ROC curve after n classifier for test data set:



K-Neighbors Map:



1.6) Model Tuning (4 pts) , Bagging (1.5 pts) and Boosting (1.5 pts). Apply grid search on each model (include all models) and make models on best_params. Define a logic behind choosing particular values for different hyper-parameters for grid search. Compare and comment on performances of all. Comment on feature importance if applicable. Successful implementation of both algorithms along with inferences and comments on the model performances.

Bagging Train:

Splitting the Data:

```
from sklearn.model_selection import train_test_split
X_train,X_test, Y_train,Y_test= train_test_split(X,Y,test_size=0.30,random_state=1)
```

Importing Bagging Classifier:

```
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
```

Fitting the Model:

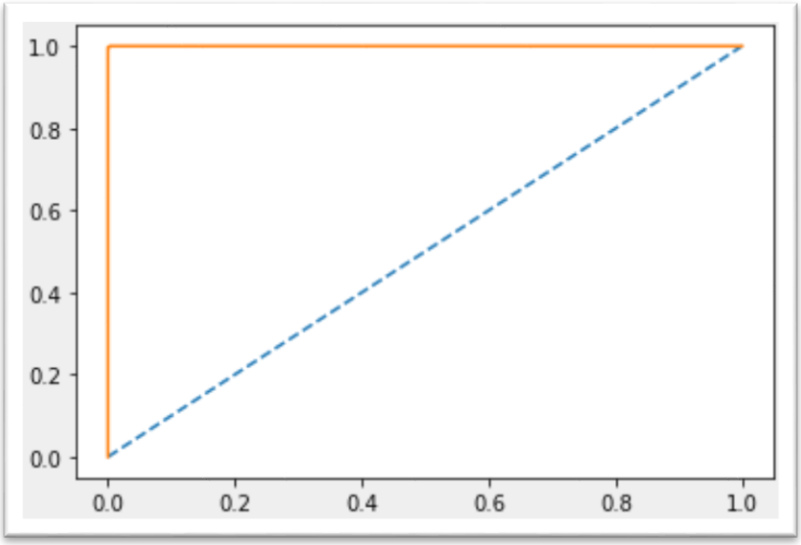
```
cart=DecisionTreeClassifier()
Bagging_model=BaggingClassifier(base_estimator=cart,n_estimators=100, random_state=1)

Bagging_model.fit(X_train,Y_train)
```

```
BaggingClassifier(base_estimator=DecisionTreeClassifier(), n_estimators=100,
                  random_state=1)
```

For Training Data Set:

```
0.9990627928772259
[[331   1]
 [  0 735]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       332
           1       1.00      1.00      1.00       735

    accuracy                           1.00      1067
   macro avg       1.00      1.00      1.00      1067
weighted avg       1.00      1.00      1.00      1067
```

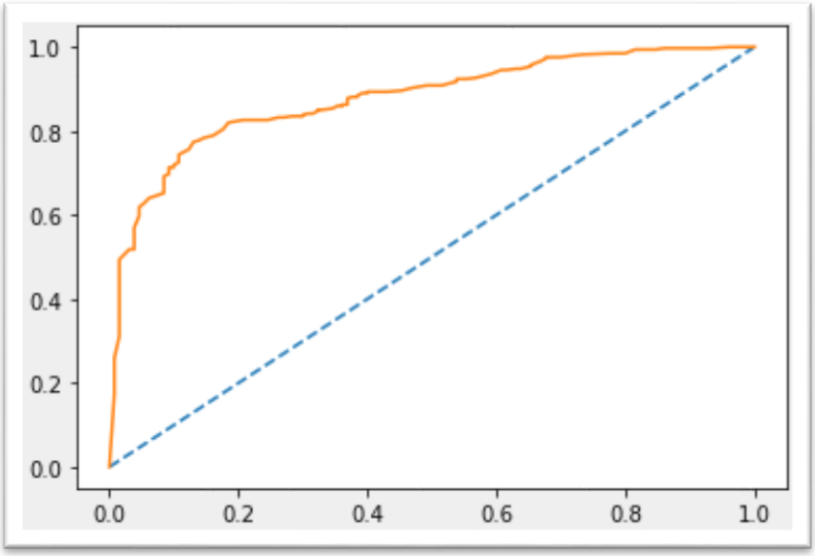
AUC _ROC Curve Bagging Train:



For Testing Data Set:

```
0.7969432314410481
[[ 83  47]
 [ 46 282]]
              precision    recall  f1-score   support

           0       0.64      0.64      0.64       130
           1       0.86      0.86      0.86       328

    accuracy                           0.80       458
   macro avg       0.75      0.75      0.75       458
weighted avg       0.80      0.80      0.80       458
```

AUC_ROC Curve Bagging Test:



Boosting Train:

Ada Boost:

Importing the classifier:

```
from sklearn.ensemble import AdaBoostClassifier
```

Fitting the Model:

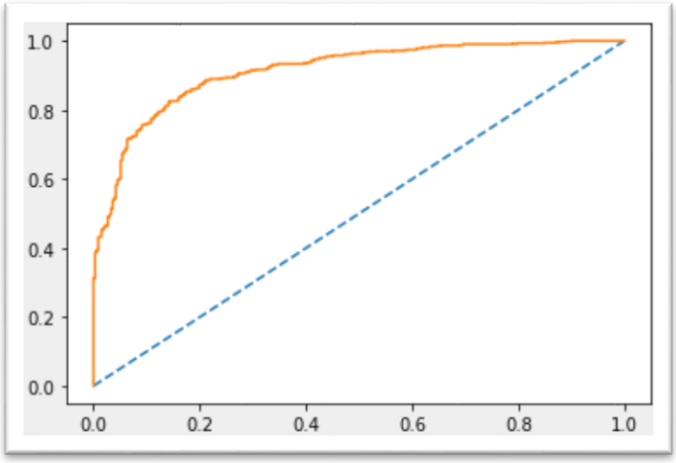
```
ADB_model=AdaBoostClassifier(n_estimators=100,random_state=1)
ADB_model.fit(X_train,Y_train)
```

```
AdaBoostClassifier(n_estimators=100, random_state=1)
```

For Training Data set:

```
0.8472352389878163
[[238  94]
 [ 69 666]]
              precision    recall  f1-score   support

           0       0.78      0.72      0.74       332
           1       0.88      0.91      0.89       735

    accuracy                           0.85      1067
   macro avg       0.83      0.81      0.82      1067
weighted avg       0.84      0.85      0.85      1067
```

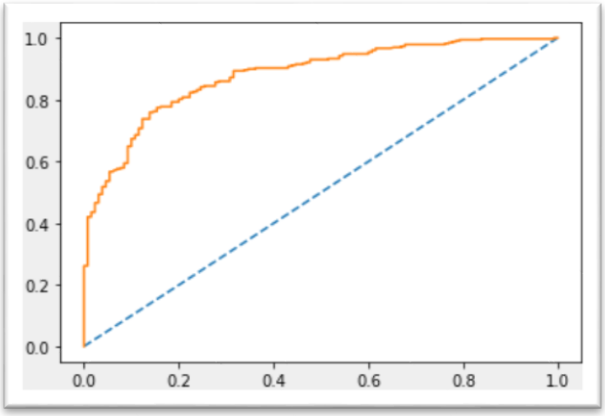
ROC Curves:



For Testing Data set:

```
0.8187772925764192
[[ 94  36]
 [ 44 284]]
              precision    recall  f1-score   support

           0       0.68      0.72      0.70       130
           1       0.89      0.87      0.88       328

    accuracy                           0.83       458
   macro avg       0.78      0.79      0.79       458
weighted avg       0.83      0.83      0.83       458
```

ROC Curves:



Gradient Boosting:

Importing and Fitting the Data Set:

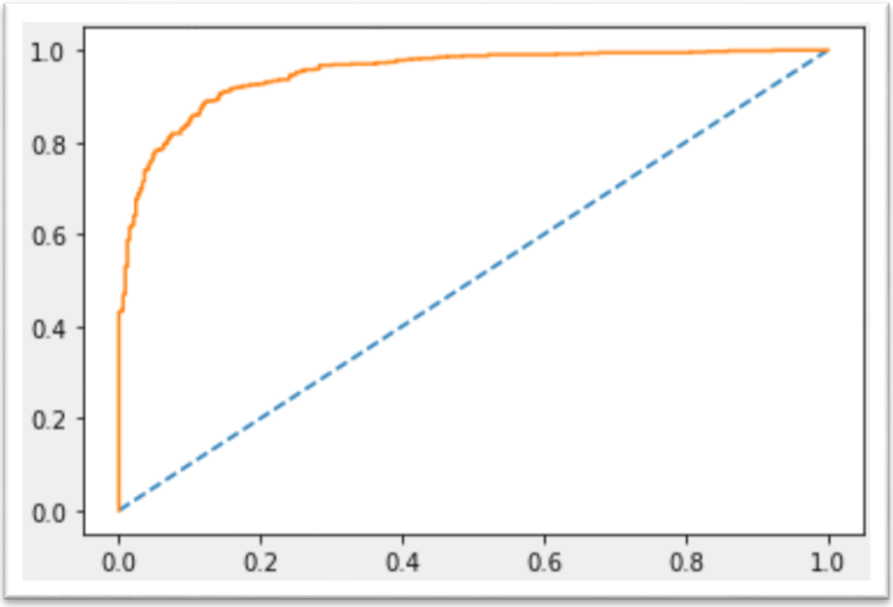
```
from sklearn.ensemble import GradientBoostingClassifier
gbc_model=GradientBoostingClassifier(random_state=1)
gbc_model.fit(X_train, Y_train)
```

```
GradientBoostingClassifier(random_state=1)
```

For Training Data set:

```
0.8865979381443299
[[240  92]
 [ 86 649]]
              precision    recall  f1-score   support

           0       0.84      0.79      0.81       332
           1       0.91      0.93      0.92       735

    accuracy                           0.89      1067
   macro avg       0.87      0.86      0.87      1067
weighted avg       0.89      0.89      0.89      1067
```

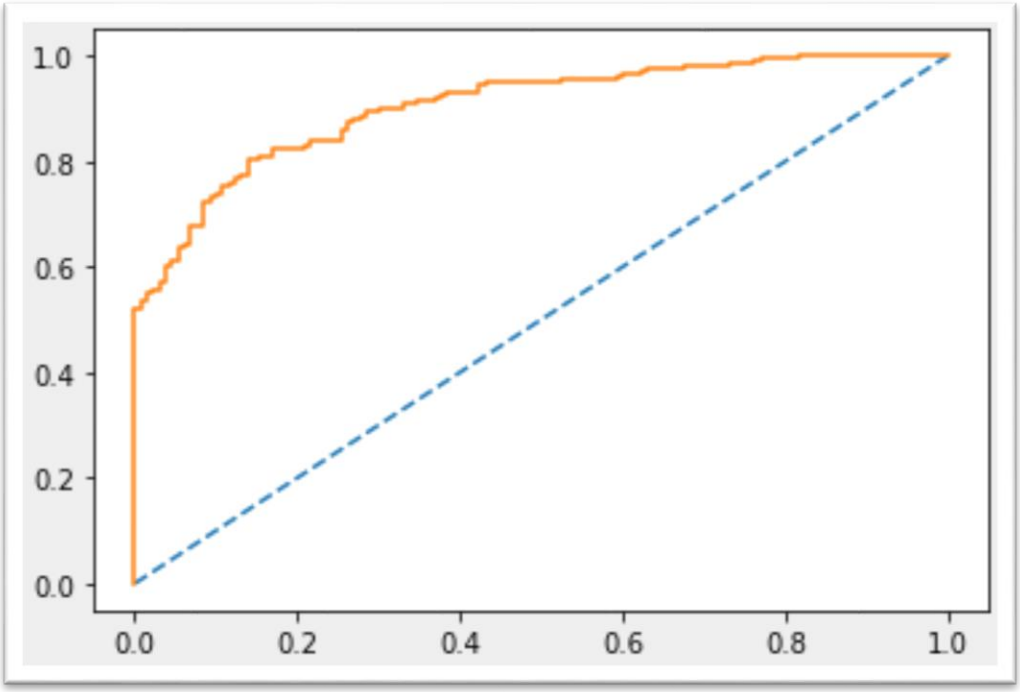
AUC _ROC Curve Boosting Train:



Gradient Boosting Test:

```
0.8318777292576419
[[ 94  36]
 [ 44 284]]
              precision    recall  f1-score   support

           0       0.68      0.72      0.70       130
           1       0.89      0.87      0.88       328

    accuracy                           0.83       458
   macro avg       0.78      0.79      0.79       458
weighted avg       0.83      0.83      0.83       458
```

Gradient Boosting AUC_ROC Curve Test:



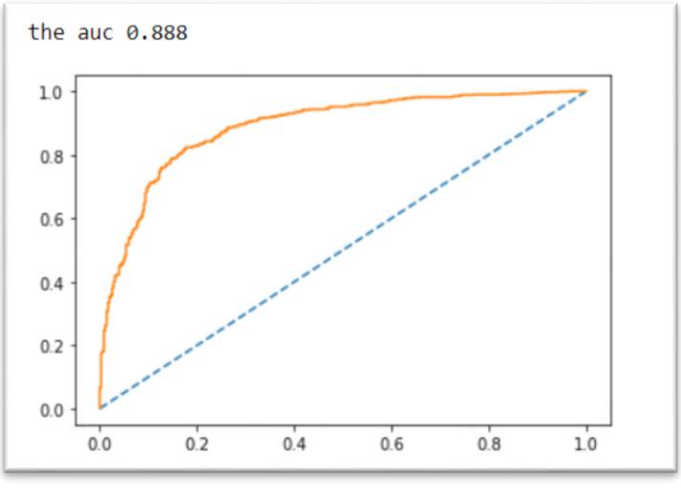
1.7 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model, classification report (4 pts) Final Model - Compare and comment on all models on the basis of the performance metrics in a structured tabular manner. Describe on which model is best/optimized, After comparison which model suits the best for the problem in hand on the basis of different measures. Comment on the final model.

Discriminant Analysis:

Training Set:

```
0.8372703412073491
[[246 105]
 [ 81 711]]
              precision    recall  f1-score   support

           0       0.75      0.70      0.73       351
           1       0.87      0.90      0.88       792

    accuracy                           0.84      1143
   macro avg       0.81      0.80      0.80      1143
weighted avg       0.83      0.84      0.84      1143
```

AUC ROC curve for LDA Train:

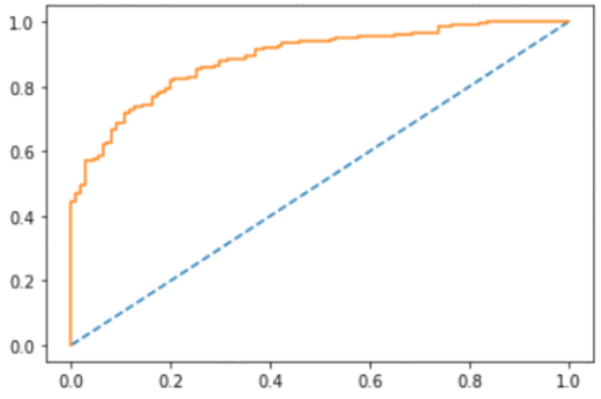


Testing Set:

```
0.824607329842932
[[ 76  35]
 [ 32 239]]
              precision    recall  f1-score   support

           0       0.70      0.68      0.69       111
           1       0.87      0.88      0.88       271

    accuracy                           0.82       382
   macro avg       0.79      0.78      0.79       382
weighted avg       0.82      0.82      0.82       382
```

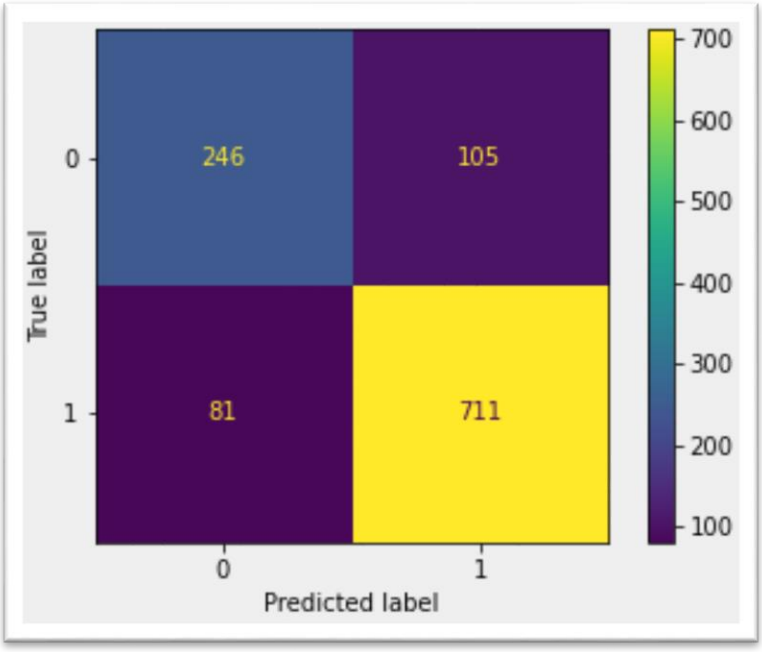
AUC ROC curve for LDA Test:

```
the auc curve 0.888

[<matplotlib.lines.Line2D at 0x254b2f003d0>]
```



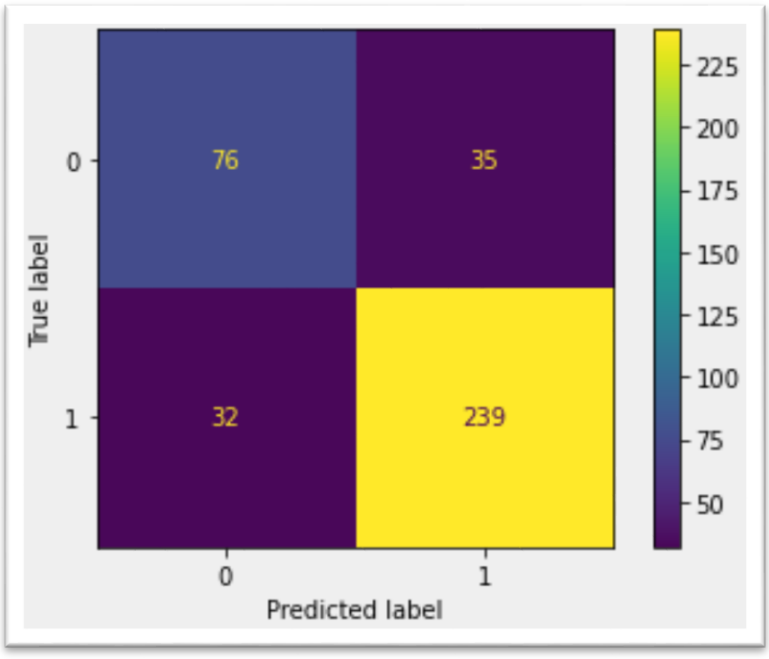
Confusion Matrix for the training data:

```
array([[242, 109],
       [ 73, 719]], dtype=int64)
```



Confusion Matrix for test data:

```
array([[ 74,  37],
       [ 31, 240]], dtype=int64)
```



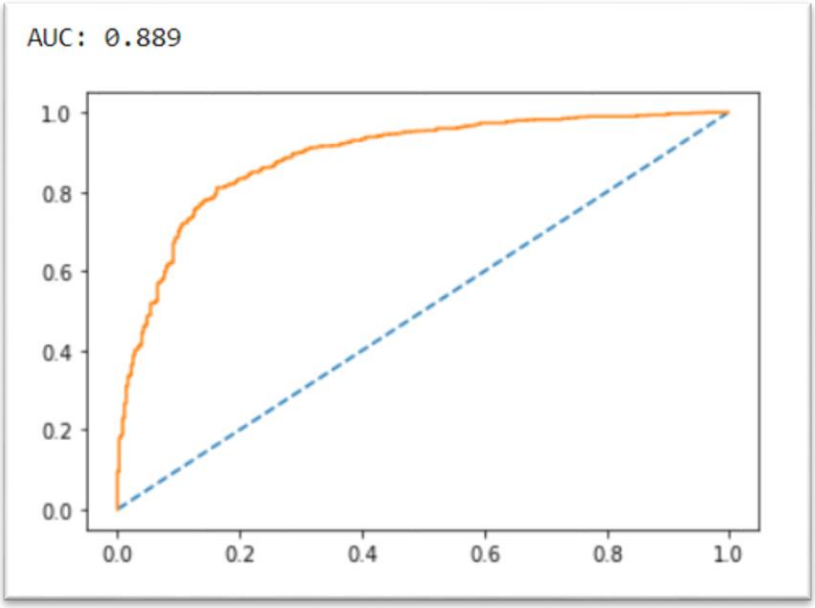
Logistic Regression:
Training Set:

```
0.8407699037620298
[[242 109]
 [ 73 719]]
              precision    recall  f1-score   support

           0       0.77      0.69      0.73       351
           1       0.87      0.91      0.89       792

    accuracy                           0.84      1143
   macro avg       0.82      0.80      0.81      1143
weighted avg       0.84      0.84      0.84      1143
```

```
Logistic_model.score(X_train,Y_train)
```

```
0.8407699037620298
```

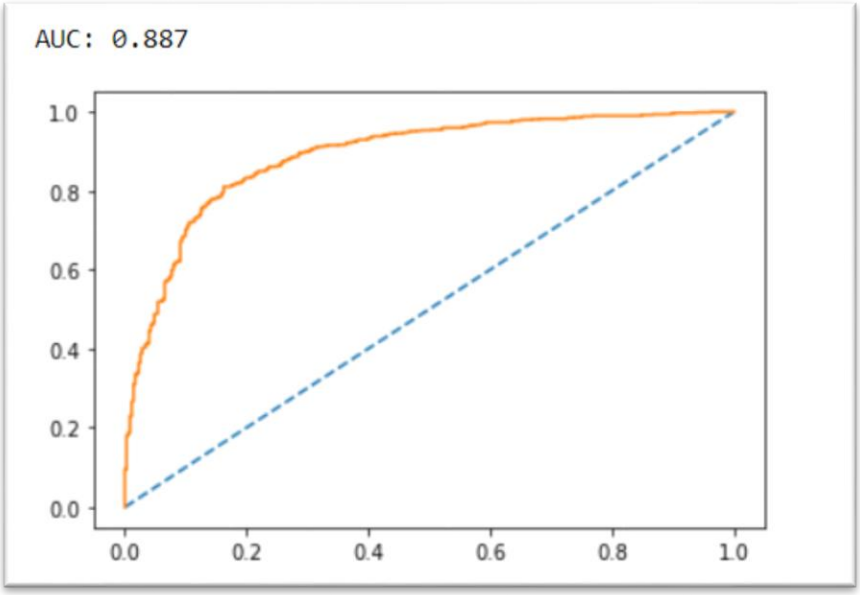
AUC ROC curve for Logistic Regression Train:



Testing Set:

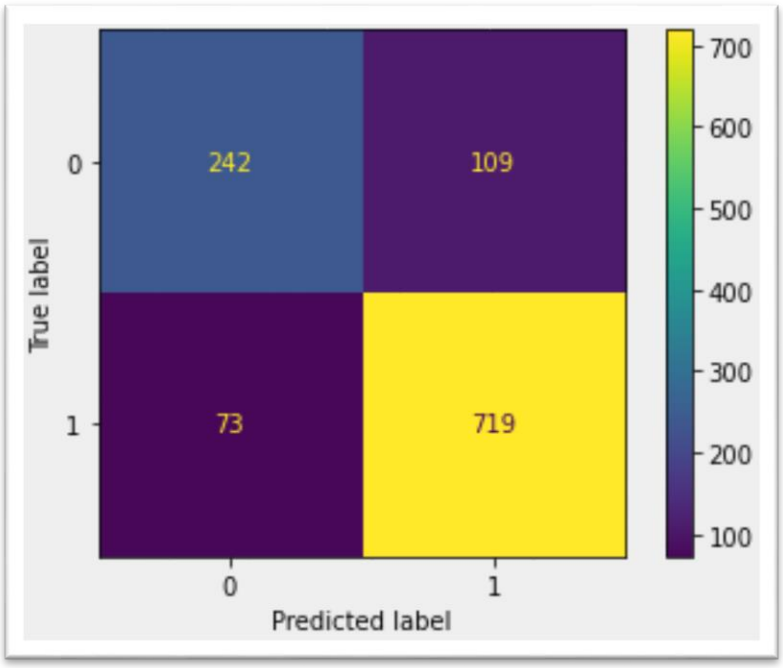
```
0.8219895287958116
[[ 74  37]
 [ 31 240]]
              precision    recall  f1-score   support

           0       0.70      0.67      0.69       111
           1       0.87      0.89      0.88       271

    accuracy                           0.82       382
   macro avg       0.79      0.78      0.78       382
weighted avg       0.82      0.82      0.82       382
```

AUC ROC curve for Logistic Regression Test:



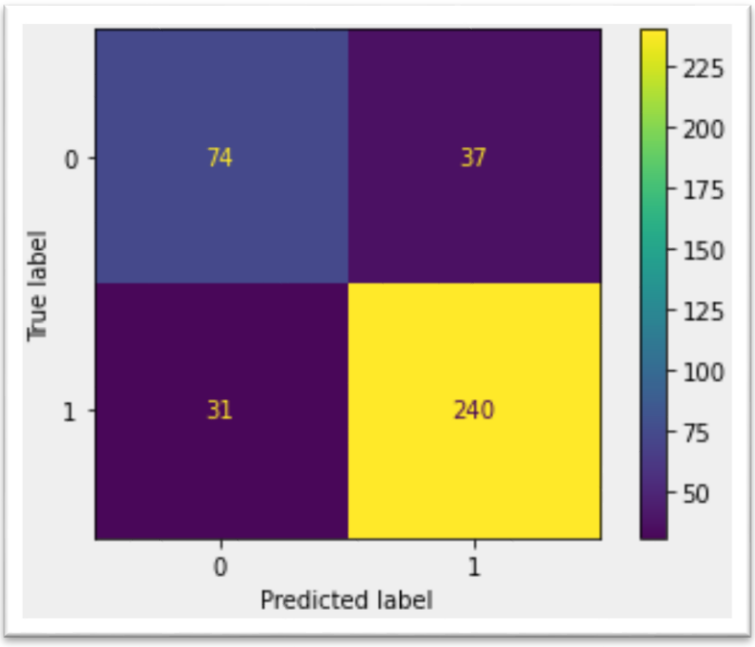
Confusion Matrix for the training data:

```
array([[242, 109],
       [ 73, 719]], dtype=int64)
```



Confusion Matrix for test data:

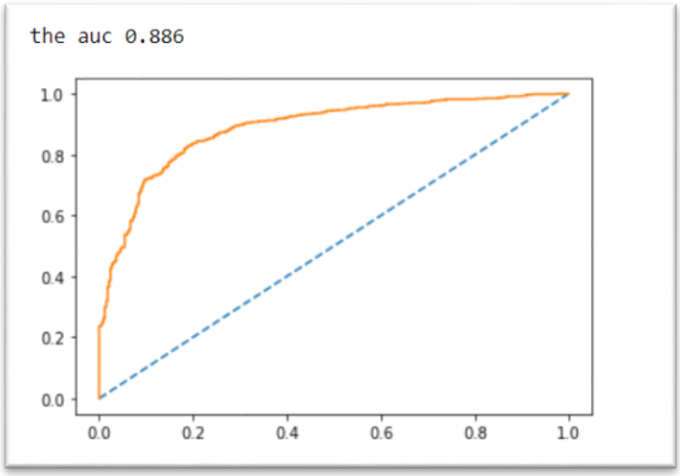
```
array([[ 74,  37],
       [ 31, 240]], dtype=int64)
```



Naive Bayes:
Training Set:

```
0.8331771321462043
[[240  92]
 [ 86 649]]
              precision    recall  f1-score   support

           0       0.74      0.72      0.73       332
           1       0.88      0.88      0.88       735

    accuracy                           0.83      1067
   macro avg       0.81      0.80      0.80      1067
weighted avg       0.83      0.83      0.83      1067
```

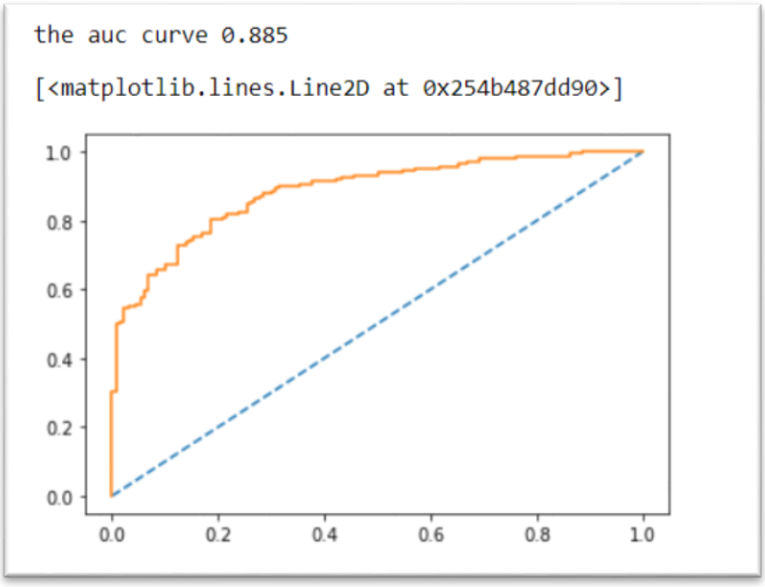
AUC Curve:



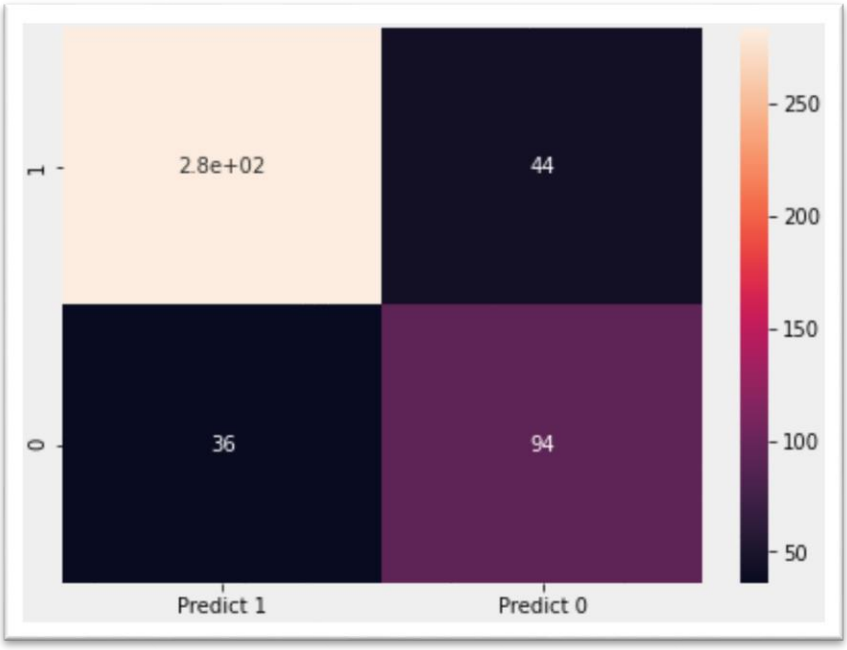
Testing Set:

```
0.8253275109170306
[[ 94  36]
 [ 44 284]]
              precision    recall  f1-score   support

           0       0.68      0.72      0.70       130
           1       0.89      0.87      0.88       328

    accuracy                           0.83       458
   macro avg       0.78      0.79      0.79       458
weighted avg       0.83      0.83      0.83       458
```

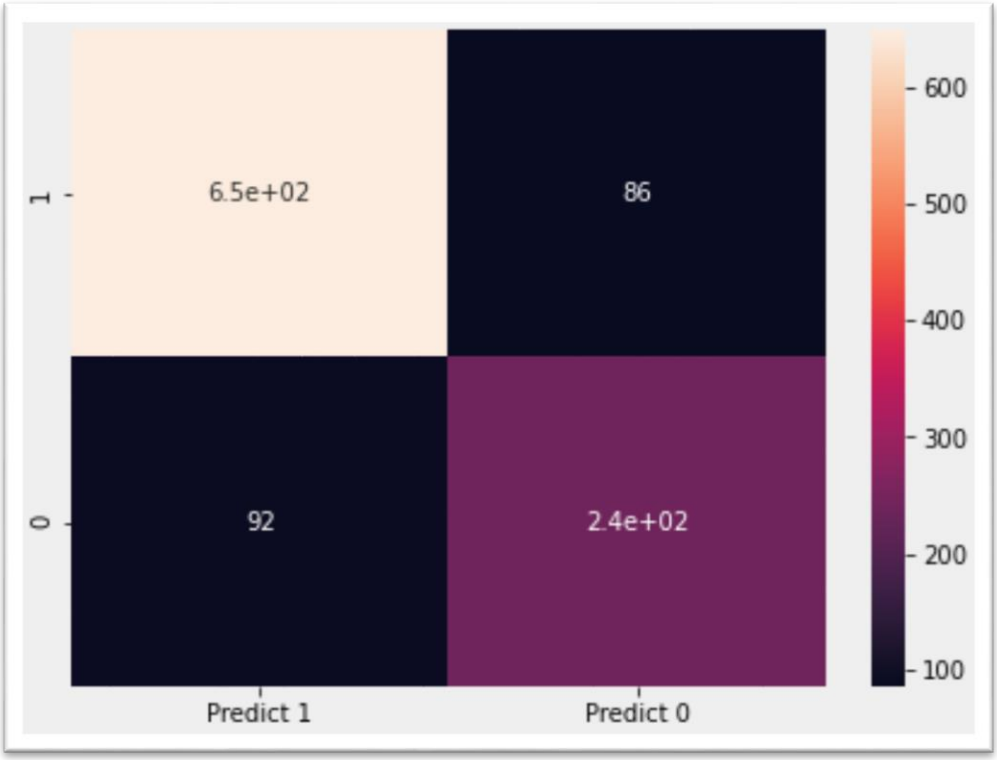
AUC Curve:



Confusion Matrix for the testing data:



Confusion Matrix for the training data:



KNN:

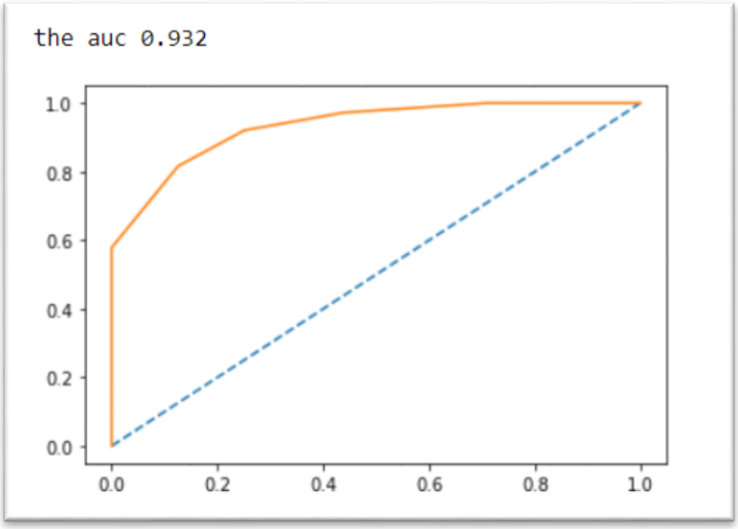
```
print(KNN_model_score)
```

```
0.8678915135608049
```

Training Set:

```
[[263  88]
 [ 63 729]]
              precision    recall  f1-score   support

           0       0.81      0.75      0.78       351
           1       0.89      0.92      0.91       792

    accuracy                           0.87      1143
   macro avg       0.85      0.83      0.84      1143
weighted avg       0.87      0.87      0.87      1143
```

AUC ROC Curve KNN Train:



Model Scores:

```
y_test_predict=KNN_model.predict(x_test)

KNN_model_score=KNN_model.score(x_test, y_test)

print(KNN_model_score)
```

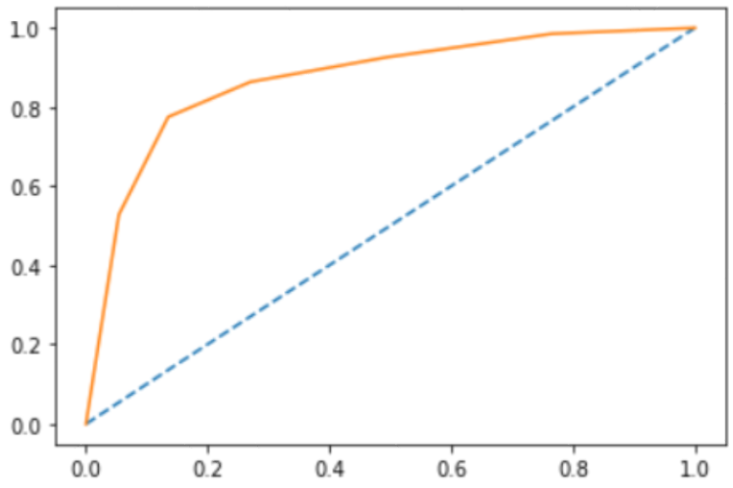
```
0.824607329842932
```

Testing Set:

```
[[ 81  30]
 [ 37 234]]
              precision    recall  f1-score   support

           0       0.69      0.73      0.71       111
           1       0.89      0.86      0.87       271

    accuracy                           0.82       382
   macro avg       0.79      0.80      0.79       382
weighted avg       0.83      0.82      0.83       382
```

AUC ROC Curve KNN Test:

```
the auc curve 0.870

[<matplotlib.lines.Line2D at 0x254b4b8ab20>]
```



For K=7

Training Set:

```
0.8530183727034121
[[253  98]
 [ 70 722]]
              precision    recall  f1-score   support

           0       0.78      0.72      0.75       351
           1       0.88      0.91      0.90       792

    accuracy                           0.85      1143
   macro avg       0.83      0.82      0.82      1143
weighted avg       0.85      0.85      0.85      1143
```

Testing Set:

```
0.8350785340314136
[[ 84  27]
 [ 36 235]]
              precision    recall  f1-score   support

           0       0.70      0.76      0.73       111
           1       0.90      0.87      0.88       271

    accuracy                           0.84       382
   macro avg       0.80      0.81      0.80       382
weighted avg       0.84      0.84      0.84       382
```

For K=5

Training Set:

```
0.8678915135608049
[[263  88]
 [ 63 729]]
              precision    recall  f1-score   support

           0       0.81      0.75      0.78       351
           1       0.89      0.92      0.91       792

    accuracy                           0.87      1143
   macro avg       0.85      0.83      0.84      1143
weighted avg       0.87      0.87      0.87      1143
```

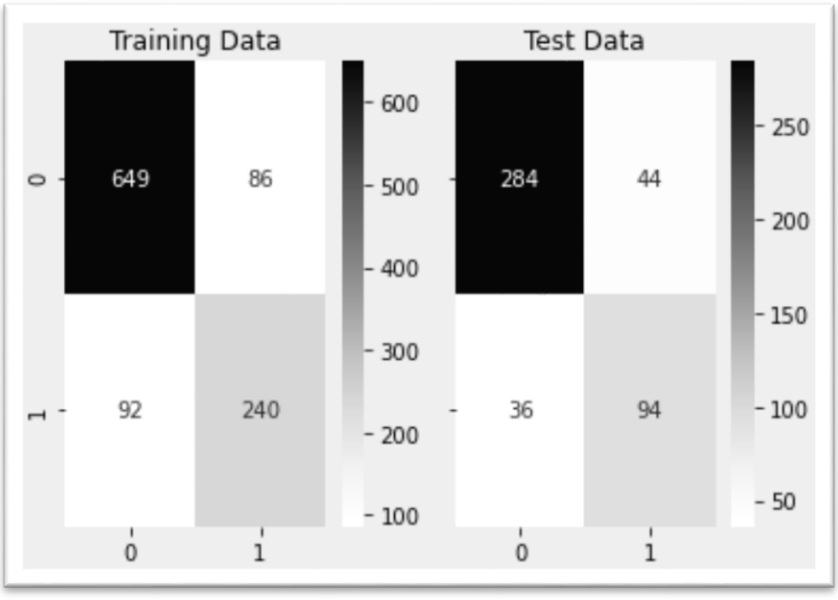
Testing Set:

```
0.824607329842932
[[ 81  30]
 [ 37 234]]
              precision    recall  f1-score   support

           0       0.69      0.73      0.71       111
           1       0.89      0.86      0.87       271

    accuracy                           0.82       382
   macro avg       0.79      0.80      0.79       382
weighted avg       0.83      0.82      0.83       382
```

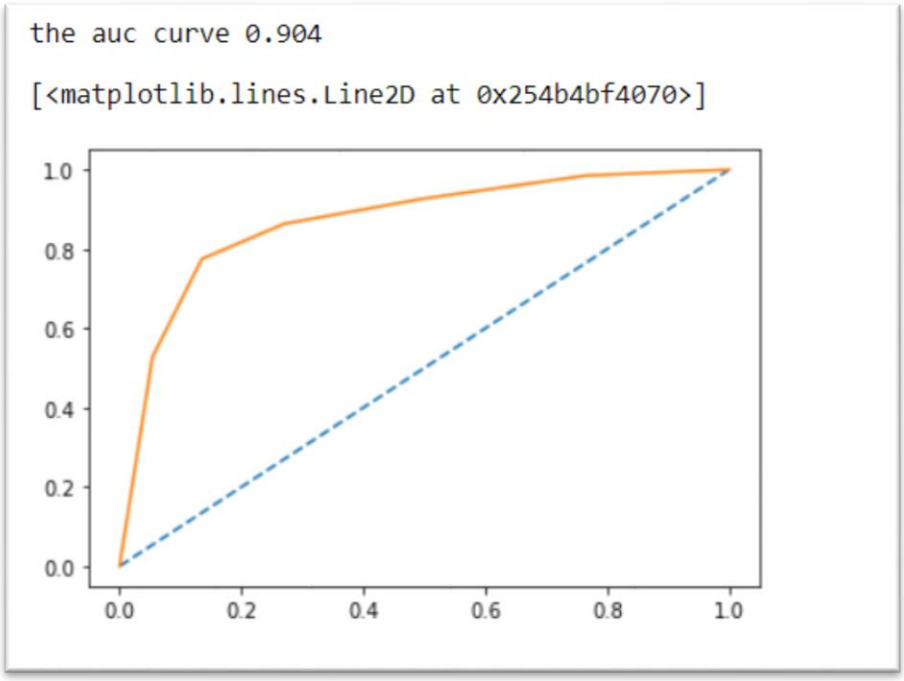
Accuracy Scores:

```
[0.23298429319371727,
 0.19633507853403143,
 0.17539267015706805,
 0.16492146596858637,
 0.17801047120418845,
 0.17277486910994766,
 0.17539267015706805,
 0.18586387434554974,
 0.17801047120418845,
 0.17277486910994766]
```

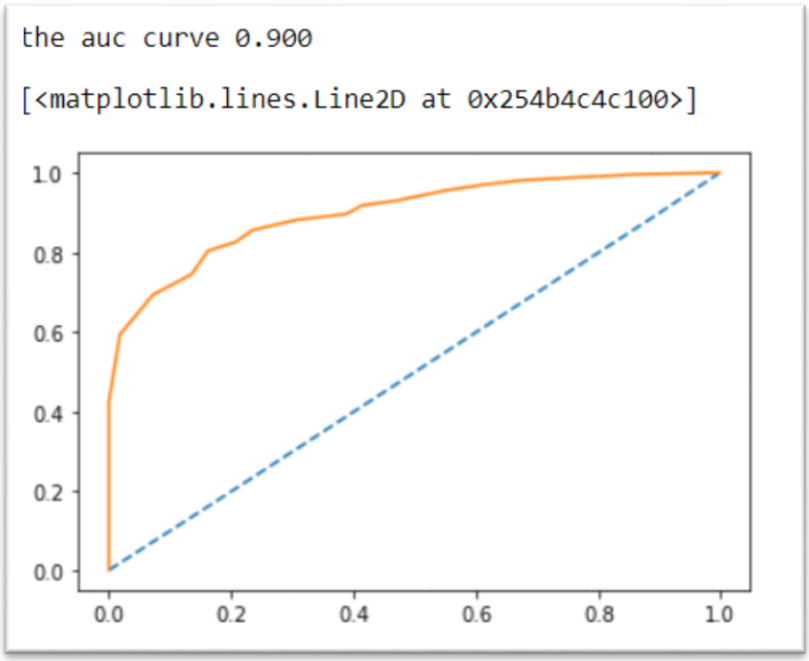
Confusion Matrix:



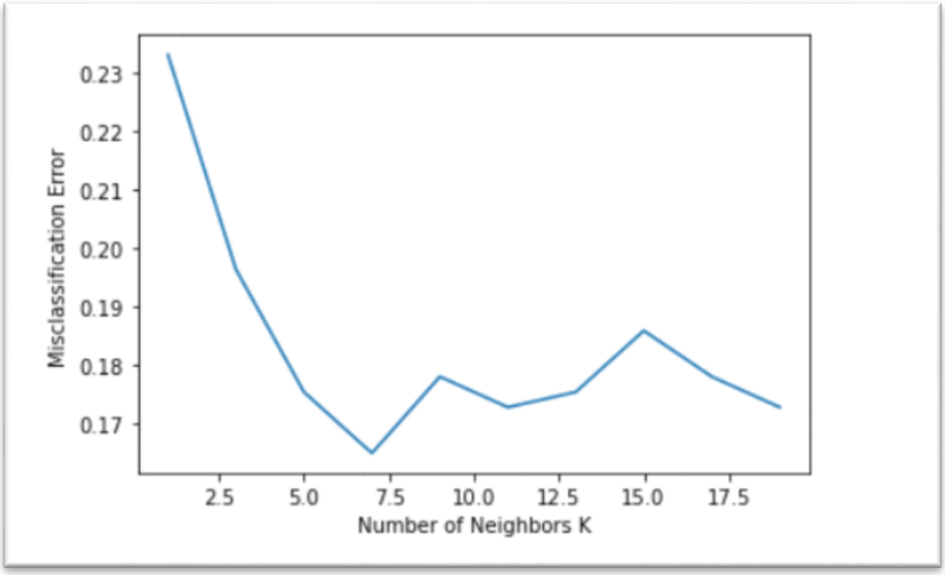
AUC ROC curve after n classifier for train data set:



AUC ROC curve after n classifier for test data set:



KNN Map:



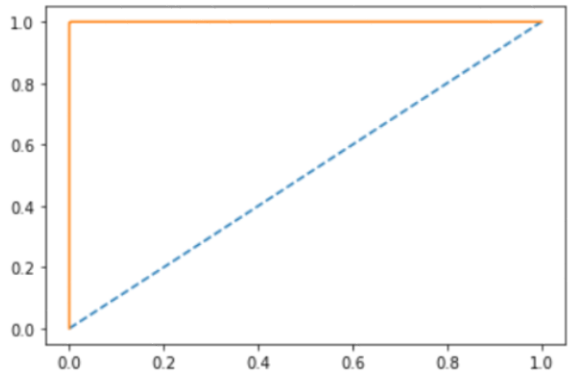
Bagging Train:

Training Set:

```
0.9990627928772259
[[331   1]
 [  0 735]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       332
           1       1.00      1.00      1.00       735

    accuracy                           1.00      1067
   macro avg       1.00      1.00      1.00      1067
weighted avg       1.00      1.00      1.00      1067
```

AUC _ROC Curve Bagging Train:

```
AUC: 1.000
[<matplotlib.lines.Line2D at 0x254b61a5a90>]
```

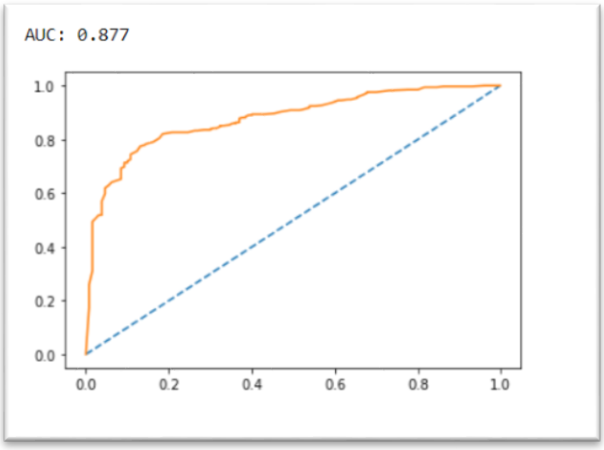


Bagging Test:

Testing Set:

```
0.7969432314410481
[[ 83  47]
 [ 46 282]]
              precision    recall  f1-score   support

           0       0.64      0.64      0.64       130
           1       0.86      0.86      0.86       328

    accuracy                           0.80       458
   macro avg       0.75      0.75      0.75       458
weighted avg       0.80      0.80      0.80       458
```

AUC _ROC Curve Bagging Test:



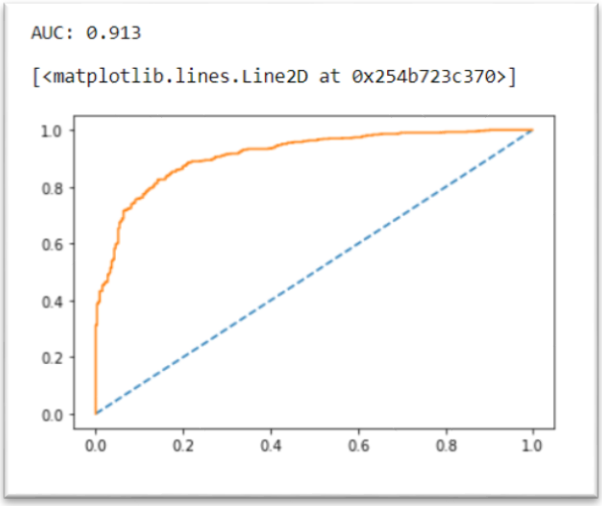
Boosting Train:

Ada Boost:

Training Set:

```
0.8472352389878163
[[238  94]
 [ 69 666]]
              precision    recall  f1-score   support

           0       0.78      0.72      0.74       332
           1       0.88      0.91      0.89       735

    accuracy                           0.85      1067
   macro avg       0.83      0.81      0.82      1067
weighted avg       0.84      0.85      0.85      1067
```

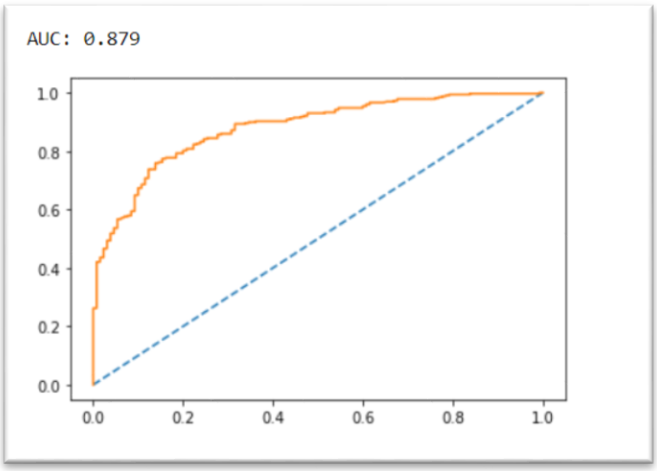
AUC Curve:



Testing Set:

```
0.8187772925764192
[[ 94  36]
 [ 44 284]]
              precision    recall  f1-score   support

           0       0.68      0.72      0.70       130
           1       0.89      0.87      0.88       328

    accuracy                           0.83       458
   macro avg       0.78      0.79      0.79       458
weighted avg       0.83      0.83      0.83       458
```

AUC Curve:



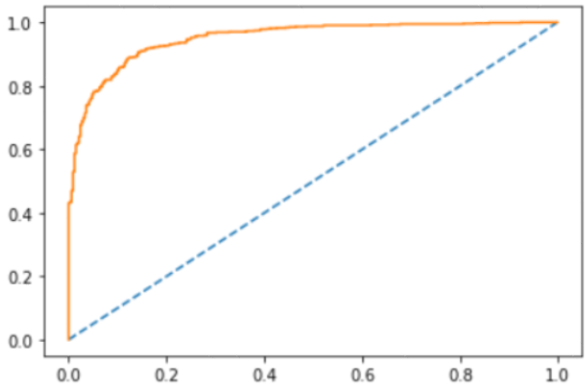
Gradient Boosting:

Training Set:

```
0.8865979381443299
[[240  92]
 [ 86 649]]
              precision    recall  f1-score   support

           0       0.84      0.79      0.81       332
           1       0.91      0.93      0.92       735

    accuracy                           0.89      1067
   macro avg       0.87      0.86      0.87      1067
weighted avg       0.89      0.89      0.89      1067
```

AUC _ROC Curve Boosting Train:

```
AUC: 0.950

[<matplotlib.lines.Line2D at 0x254b729f9a0>]
```

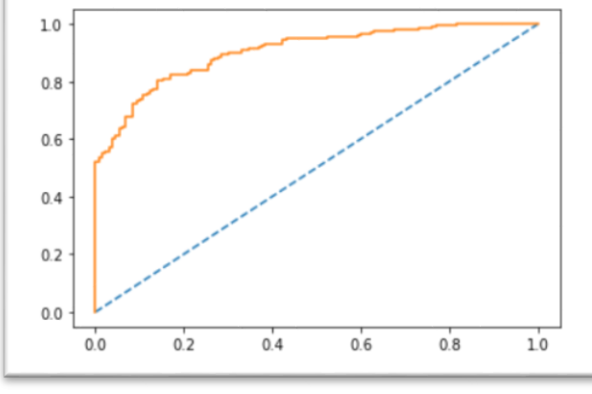


Testing Set:

```
0.8318777292576419
[[ 94  36]
 [ 44 284]]
              precision    recall  f1-score   support

           0       0.68      0.72      0.70       130
           1       0.89      0.87      0.88       328

    accuracy                           0.83       458
   macro avg       0.78      0.79      0.79       458
weighted avg       0.83      0.83      0.83       458
```

AUC Curve

```
AUC: 0.904

[<matplotlib.lines.Line2D at 0x254b5ea3a90>]
```



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. There should be at least 3-4 Recommendations and insights in total. Recommendations should be easily understandable and business specific, students should not give any technical suggestions. Full marks should only be allotted if the recommendations are correct and business specific.

Inferences

- Logistic Regression performed the best out of all the models build.
- Logistic Regression Equation for the model:
  (3.05008) * Intercept + (-0.01891) * age + (0.41855) * economic_cond_national + (0.06714) * economic_cond_household + (0.62627) * Blair + (-0.83974) * Hague + (- 0.21413) * Europe + (-0.40331) * political_knowledge + (0.10881) * gender

The above equation help in understanding the model and the feature importance, how each feature contributes to the predicted output.

Top 5 features in Logistic Regression Model in order of decreasing importance are-

1. Hague : |-0.8181846212178241|
2. Blair : |0.5460018962250501|
3. economic_cond_national : |0.37700497490783885|
4. political_knowledge : |-0.3459485608005413|
5. Europe : |-0.19691071679312278|

Insights and Recommendations

Our main Business Objective is - "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."

- Using Logistic Regression Model without scaling for predicting the outcome as it has the best optimised performance.
- Hyper-parameters tuning is an important aspect of model building. There are limitations to this as to process these combinations huge amount of processing power is required. But if tuning can be done with many sets of parameters than we might get even better results.
- Gathering more data will also help in training the models and thus improving their predictive powers.
- Boosting Models can also perform well like CATBoost performed well even without tuning. Thus, if we perform hyper-parameters tuning we might get better results.
- We can also create a function in which all the models predict the outcome in sequence. This will helps in better understanding and the probability of what the outcome will be.

Problem 2:

Executive Summary:

The given data set has number columns and rows which is having the speech given by 3 Presidents. In this, I had made 3 Excel files for each President and then have made use of Text Analysis in order to fetch the desired outcomes from the same.

Background of the problem:

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

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

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

President Franklin D. Roosevelt in 1941:

```
df1.head()
```

	Name	Speech
0	Roosevelt	On each national day of inauguration since 178...

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1 entries, 0 to 0
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Name    1 non-null      object
 1   Speech  1 non-null      object
dtypes: object(2)
memory usage: 144.0+ bytes
```

```
all_Words=[x for x in pd.Series(' '.join(df1['Speech']).split())]
```

Most Common words before removing stop words:

```
[('the', 104),
 ('of', 81),
 ('and', 41),
 ('to', 35),
 ('in', 30),
 ('a', 28),
 ('is', 24),
 ('--', 22),
 ('we', 22),
 ('that', 21)]
```

Number of words:

	Speech	totalwords
0	On each national day of inauguration since 178...	1323

Number of Characters- including spaces:

	Speech	char_count
0	On each national day of inauguration since 178...	7651

```
df1['Speech']
```

```
0    On each national day of inauguration since 178...
Name: Speech, dtype: object
```

```
len(df1['Speech'])
```

```
1
```

Average Word Length:

	Speech	avg_word
0	On each national day of inauguration since 178...	4.783825

Number of stop Words:

	Speech	stopwords
0	On each national day of inauguration since 178...	632

Number of special character:

	Speech	hastags
0	On each national day of inauguration since 178...	0

Number of Numeric:

	Speech	numerics
0	On each national day of inauguration since 178...	14

Number of Uppercase Words:

	Speech	upper
0	On each national day of inauguration since 178...	1

Number of Uppercase Letters:

	Speech	upper_letter
0	On each national day of inauguration since 178...	119

Lower Case conversion:

```
df1['Speech'] = df1['Speech'].apply(lambda x: " ".join(x.lower() for x in x.split()))
df1['Speech'].head()
```

```
0    on each national day of inauguration since 178...
Name: Speech, dtype: object
```

President John F. Kennedy in 1961

	Name	Speech
0	Kennedy	Vice President Johnson, Mr. Speaker, Mr. Chief...

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1 entries, 0 to 0
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Name    1 non-null      object
 1   Speech  1 non-null      object
dtypes: object(2)
memory usage: 144.0+ bytes
```

```
all_Words=[x for x in pd.Series(' '.join(df2['Speech']).split())]
```

Most Common words before removing stop words:

```
[('the', 83),
 ('of', 65),
 ('and', 37),
 ('to', 37),
 ('a', 29),
 ('we', 27),
 ('--', 24),
 ('in', 24),
 ('our', 21),
 ('not', 19)]
```

Number of words:

	Speech	totalwords
0	Vice President Johnson, Mr. Speaker, Mr. Chief...	1364

Number of Characters- including spaces:

	Speech	char_count
0	Vice President Johnson, Mr. Speaker, Mr. Chief...	7673

```
df2['Speech']
```

```
0    Vice President Johnson, Mr. Speaker, Mr. Chief...
Name: Speech, dtype: object
```

```
len(df2['Speech'])
```

```
1
```

Average Word Length:

	Speech	avg_word
0	Vice President Johnson, Mr. Speaker, Mr. Chief...	4.6261

Number of stop Words:

	Speech	stopwords
0	Vice President Johnson, Mr. Speaker, Mr. Chief...	618

Number of special character:

	Speech	hastags
0	Vice President Johnson, Mr. Speaker, Mr. Chief...	0

Number of Numeric:

	Speech	numerics
0	Vice President Johnson, Mr. Speaker, Mr. Chief...	7

Number of Uppercase Words:

	Speech	upper
0	Vice President Johnson, Mr. Speaker, Mr. Chief...	5

Number of Uppercase Letters:

	Speech	upper_letter
0	Vice President Johnson, Mr. Speaker, Mr. Chief...	94

Lower Case conversion:

```
df2['Speech'] = df2['Speech'].apply(lambda x: " ".join(x.lower() for x in x.split()))
df2['Speech'].head()
```

```
0    vice president johnson, mr. speaker, mr. chief...
Name: Speech, dtype: object
```

President Richard Nixon in 1973:

	Name	Speech
0	Nixon	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1 entries, 0 to 0
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Name    1 non-null      object
 1   Speech  1 non-null      object
dtypes: object(2)
memory usage: 144.0+ bytes
```

```
all_Words=[x for x in pd.Series(' '.join(df3['Speech']).split())]
```

Most Common words before removing stop words:

```
[('the', 80),
 ('of', 68),
 ('to', 65),
 ('in', 54),
 ('and', 47),
 ('we', 38),
 ('a', 34),
 ('that', 32),
 ('for', 32),
 ('our', 31)]
```

Number of words:

	Speech	totalwords
0	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	1769

Number of Characters- including spaces:

	Speech	char_count
0	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	10106

```
df3['Speech']
```

```
0    Mr. Vice President, Mr. Speaker, Mr. Chief Jus...
Name: Speech, dtype: object
```

```
len(df3['Speech'])
```

```
1
```

Average Word Length:

	Speech	avg_word
0	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	4.713397

Number of stop Words:

	Speech	stopwords
0	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	899

Number of special character:

	Speech	hastags
0	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	0

Number of Numeric:

	Speech	numerics
0	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	10

Number of Uppercase Words:

	Speech	upper
0	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	13

Number of Uppercase Letters:

	Speech	upper_letter
0	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	132

Lower Case conversion:

```
df3['Speech'] = df3['Speech'].apply(lambda x: " ".join(x.lower() for x in x.split()))
df3['Speech'].head()
```

```
0    mr. vice president, mr. speaker, mr. chief jus...
Name: Speech, dtype: object
```

2.2 Remove all the stopwords from the three speeches. Show the word count before and after the removal of stopwords. Show a sample sentence after the removal of stopwords.

President Franklin D. Roosevelt in 1941:
Most Common words before removing stop words:

```
[('the', 104),
 ('of', 81),
 ('and', 41),
 ('to', 35),
 ('in', 30),
 ('a', 28),
 ('is', 24),
 ('--', 22),
 ('we', 22),
 ('that', 21)]
```

Number of word count:

	Speech	totalwords
0	On each national day of inauguration since 178...	1323

Number of stop words:

	Speech	stopwords
0	On each national day of inauguration since 178...	632

Word Count after removing stop words:

	Speech	totalwords
0	national day inauguration since 1789 people re...	624

Removal of stop words:

```
from nltk.corpus import stopwords
stop = stopwords.words('english')
df1['Speech'] = df1['Speech'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
df1['Speech'].head()
```

```
0    national day inauguration since 1789 people re...
Name: Speech, dtype: object
```

Most Common words after removing stop words:

```
know        10
nation      10
spirit       8
us           8
life         7
america      6
years        6
people       6
democracy    6
speaks       5
dtype: int64
```

Rare Words Removal:

```
well         1
knowledge    1
alien        1
plain        1
live         1
matters      1
common       1
spreading    1
purposes     1
threeway     1
dtype: int64
```

Stemming refers to the removal of suffices, like “ing”, “ly”, “s”, etc. by a simple rule-based approach:

```
from nltk.stem import PorterStemmer
st = PorterStemmer()
df1['Speech'].apply(lambda x: " ".join([st.stem(word) for word in x.split()]))
```

```
0    nation day inaugur sinc 1789 peopl renew sens ...
Name: Speech, dtype: object
```

Sample Passage:

```
df1['Speech'][0]  ## sentences after removing the stops words:
```

```
'national day inauguration since 1789 people renewed sense dedication united statesnnin washingtons day task people create weld
together nationnnin lincolns day task people preserve nation disruption withinnnin day task people save nation institutions dis
ruption withoutnnto us come time midst swift happenings pause moment take stock recall place history rediscover may risk real p
eril inactionnnlives nations determined count years lifetime human spirit life man threescore years ten little little less life
nation fullness measure livennthere men doubt men believe democracy form government frame life limited measured kind mystical a
rtificial fate unexplained reason tyranny slavery become surging wave future freedom ebbing tidennbut americans know truenneigh
t years ago life republic seemed frozen fatalistic terror proved true midst shock acted acted quickly boldly decisivelynnthese
later years living years fruitful years people democracy brought us greater security hope better understanding lifes ideals mea
sured material thingsnnmost vital present future experience democracy successfully survived crisis home put away many evil thin
gs built new structures enduring lines maintained fact democracynnfor action taken within threeway framework constitution unite
d states coordinate branches government continue freely function bill rights remains inviolate freedom elections wholly maintai
ned prophets downfall american democracy seen dire predictions come naughtnndemocracy dyingnnwe know seen reviveand grownnwe kn
ow cannot die built unhampered initiative individual men women joined together common enterprise enterprise undertaken carried
free expression free majoritynnwe know democracy alone forms government enlists full force mens enlightened willnnwe know democ
racy alone constructed unlimited civilization capable infinite progress improvement human lifennwe know look surface sense stil
l spreading every continent humane advanced end unconquerable forms human societynna nation like person bodya body must fed clo
thed housed invigorated rested manner measures objectives timenna nation like person mind mind must kept informed alert must kn
ow understands hopes needs neighbors nations live within narrowing circle worldnnand nation like person something deeper someth
ing permanent something larger sum parts something matters future calls forth sacred guarding presentnnit thing find difficult
even impossible hit upon single simple wordnnand yet understand spirit faith america product centuries born multitudes came man
y lands high degree mostly plain people sought early late find freedom freelynnthe democratic aspiration mere recent phase huma
n history human history permeated ancient life early peoples blazed anew middle ages written magna chartannin americas impact i
rresistible america new world tongues peoples continent newfound land came believed could create upon continent new life life n
ew freedomnnits vitality written mayflower compact declaration independence constitution united states gettysburg addressnnthos
e first came carry longings spirit millions followed stock sprang moved forward constantly consistently toward ideal gained sta
ture clarity generationnnthe hopes republic cannot forever tolerate either undeserved poverty selfserving wealthnnwe know still
far go must greatly build security opportunity knowledge every citizen measure justified resources capacity landnnbut enough ac
hieve purposes alone enough clothe feed body nation instruct inform mind also spirit three greatest spiritnnwithout body mind m
en know nation could livennbut spirit america killed even though nations body mind constricted alien world lived america know w
ould perishednnthat spirit faith speaks us daily lives ways often unnoticed seem obvious speaks us capital nation speaks us pro
cesses governing sovereignties 48 states speaks us counties cities towns villages speaks us nations hemisphere across seas ensl
aved well free sometimes fail hear heed voices freedom us privilege freedom old old storynnthe destiny america proclaimed words
prophecy spoken first president first inaugural 1789 words almost directed would seem year 1941 preservation sacred fire libert
y destiny republican model government justly considered deeply finally staked experiment intrusted hands american peoplennif lo
se sacred fireif let smothered doubt fear shall reject destiny washington strove valiantly triumphantly establish preservation
```

President John F. Kennedy in 1961:
Most Common words before removing stop words:

```
[('the', 83),
 ('of', 65),
 ('and', 37),
 ('to', 37),
 ('a', 29),
 ('we', 27),
 ('--', 24),
 ('in', 24),
 ('our', 21),
 ('not', 19)]
```

Number of word count:

	Speech	totalwords
0	Vice President Johnson, Mr. Speaker, Mr. Chief...	1364

Number of stop words:

	Speech	stopwords
0	Vice President Johnson, Mr. Speaker, Mr. Chief...	618

Removal of stop words:

```
from nltk.corpus import stopwords
stop = stopwords.words('english')
df2['Speech'] = df2['Speech'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
df2['Speech'].head()
```

```
0    vice president johnson mr speaker mr chief jus...
Name: Speech, dtype: object
```

Word Count after removing stop words:

	Speech	totalwords
0	vice president johnson mr speaker mr chief jus...	689

Most Common words after removing stop words:

```
us          11
let         11
sides        8
pledge       7
new          7
world        6
ask          5
power        5
shall        5
citizens     5
dtype: int64
```

Rare Words Removal:

```
support      1
destroy      1
eisenhower   1
subject      1
civility     1
request      1
corners      1
judge        1
mankinds     1
hard         1
dtype: int64
```

Stemming refers to the removal of suffices, like “ing”, “ly”, “s”, etc. by a simple rule-based approach:

```
from nltk.stem import PorterStemmer
st = PorterStemmer()
df2['Speech'].apply(lambda x: " ".join([st.stem(word) for word in x.split()]))
```

```
0    vice presid johnson mr speaker mr chief justic...
Name: Speech, dtype: object
```

Sample Passage:

```
df2['Speech'][0]  ## sentences after removing the stops words:
```

'vice president johnson mr speaker mr chief justice president eisenhower vice president nixon president truman reverend clergy fellow citizens observe today victory party celebration freedom symbolizing end well beginning signifying renewal well change s worn almighty god solemn oath forebears l prescribed nearly century three quarters agonnthe world different man holds mortal ha nds power abolish forms human poverty forms human life yet revolutionary beliefs forebears fought still issue around globe beli ef rights man come generosity state hand godnnwe dare forget today heirs first revolution let word go forth time place friend f oe alike torch passed new generation americans born century tempered war disciplined hard bitter peace proud ancient heritage u nwilling witness permit slow undoing human rights nation always committed committed today home around worldnnlet every nation k now whether wishes us well ill shall pay price bear burden meet hardship support friend oppose foe order assure survival succes s libertynnthis much pledge morennto old allies whose cultural spiritual origins share pledge loyalty faithful friends united l ittle cannot host cooperative ventures divided little dare meet powerful challenge odds split asundernnto new states welcome ra nks free pledge word one form colonial control shall passed away merely replaced far iron tyranny shall always expect find supp orting view shall always hope find strongly supporting freedom remember past foolishly sought power riding back tiger ended ins idennto peoples huts villages across globe struggling break bonds mass misery pledge best efforts help help whatever period req uired communists may seek votes right free society cannot help many poor cannot save richnnto sister republics south border off er special pledge convert good words good deeds new alliance progress assist free men free governments casting chains poverty p eaceful revolution hope cannot become prey hostile powers let neighbors know shall join oppose aggression subversion anywhere a mericas let every power know hemisphere intends remain master housennto world assembly sovereign states united nations last bes t hope age instruments war far outpaced instruments peace renew pledge supportto prevent becoming merely forum invective streng then shield new weak enlarge area writ may runnnfinally nations would make adversary offer pledge request sides begin anew ques t peace dark powers destruction unleashed science engulf humanity planned accidental selfdestructionnnwe dare tempt weakness ar ms sufficient beyond doubt certain beyond doubt never employednnbut neither two great powerful groups nations take comfort pres ent course sides overburdened cost modern weapons rightly alarmed steady spread deadly atom yet racing alter uncertain balance terror stays hand mankinds final warnnso let us begin anew remembering sides civility sign weakness sincerity always subject pr oof let us never negotiate fear let us never fear negotiatennlet sides explore problems unite us instead belaboring problems di vide usnnlet sides first time formulate serious precise proposals inspection control arms bring absolute power destroy nations absolute control nationsnnlet sides seek invoke wonders science instead terrors together let us explore stars conquer deserts e radicate disease tap ocean depths encourage arts commercennlet sides unite heed corners earth command isaiah undo heavy burdens let oppressed go freennand beachhead cooperation may push back jungle suspicion let sides join creating new endeavor new balanc e power new world law strong weak secure peace preservednnall finished first 100 days finished first 1000 days life administrat ion even perhaps lifetime planet let us beginnnin hands fellow citizens mine rest final success failure course since country fo unded generation americans summoned give testimony national loyalty graves young americans answered call service surround globe

President Richard Nixon in 1973:
Most Common words before removing stop words:

```
[('the', 80),
 ('of', 68),
 ('to', 65),
 ('in', 54),
 ('and', 47),
 ('we', 38),
 ('a', 34),
 ('that', 32),
 ('for', 32),
 ('our', 31)]
```

Number of word count:

	Speech	totalwords
0	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	1769

Number of stop words:

	Speech	stopwords
0	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	899

Removal of stop words:

```
from nltk.corpus import stopwords
stop = stopwords.words('english')
df3['Speech'] = df3['Speech'].apply(lambda x: " ".join(x for x in x.split() if x not in stop))
df3['Speech'].head()
```

```
0    mr vice president mr speaker mr chief justice ...
Name: Speech, dtype: object
```

Word Count after removing stop words:

	Speech	totalwords
0	mr vice president mr speaker mr chief justice ...	819

Most Common words after removing stop words:

```
us                26
peace             15
new               15
let               13
america           12
world             10
responsibility    10
great              9
government         9
shall              7
dtype: int64
```

Rare Words Removal:

```
impose          1
steadfastness   1
built           1
base            1
myselfnnin      1
else            1
rights          1
attempting      1
expectations    1
gods            1
dtype: int64
```

Stemming refers to the removal of suffices, like “ing”, “ly”, “s”, etc. by a simple rule-based approach:

```
from nltk.stem import PorterStemmer
st = PorterStemmer()
df3['Speech'].apply(lambda x: " ".join([st.stem(word) for word in x.split()]))
```

```
0    mr vice presid mr speaker mr chief justic sena...
Name: Speech, dtype: object
```

Sample Passage:

```
df3['Speech'][0]  ## sentences after removing the stops words:
```

'mr vice president mr speaker mr chief justice senator cook mrs eisenhower fellow citizens great good country share togethernnw
hen met four years ago america bleak spirit depressed prospect seemingly endless war abroad destructive conflict homennas meet
today stand threshold new era peace worldnnthe central question us shall use peace let us resolve era enter postwar periods oft
en time retreat isolation leads stagnation home invites new danger abroadnnlet us resolve become time great responsibilities gr
eatly borne renew spirit promise america enter third century nationnnthis past year saw farreaching results new policies peace
continuing revitalize traditional friendships missions peking moscow able establish base new durable pattern relationships amon
g nations world americas bold initiatives 1972 long remembered year greatest progress since end world war ii toward lasting pea
ce worldnnthe peace seek world flimsy peace merely interlude wars peace endure generations comennit important understand necess
ity limitations americas role maintaining peacennunless america work preserve peace peacennunless america work preserve freedom
freedomnnbut let us clearly understand new nature americas role result new policies adopted past four yearsnnwe shall respect t
reaty commitmentsnnwe shall support vigorously principle country right impose rule another forcennwe shall continue era negotia
tion work limitation nuclear arms reduce danger confrontation great powersnnwe shall share defending peace freedom world shall
expect others sharennthe time passed america make every nations conflict make every nations future responsibility presume tell
people nations manage affairsnnjust respect right nation determine future also recognize responsibility nation secure futurennj
ust americas role indispensable preserving worlds peace nations role indispensable preserving peacenntogether rest world let us
resolve move forward beginnings made let us continue bring walls hostility divided world long build place bridges understanding
despite profound differences systems government people world friendsnnlet us build structure peace world weak safe strong respe
cts right live different system would influence others strength ideas force armsnnlet us accept high responsibility burden glad
ly gladly chance build peace noblest endeavor nation engage gladly also act greatly meeting responsibilities abroad remain grea
t nation remain great nation act greatly meeting challenges homennwe chance today ever history make life better america ensure
better education better health better housing better transportation cleaner environment restore respect law make communities li
vable insure godgiven right every american full equal opportunitynnbecause range needs great reach opportunities great let us b
old determination meet needs new waysnnjust building structure peace abroad required turning away old policies failed building
new era progress home requires turning away old policies failednnabroad shift old policies new retreat responsibilities better
way peacennand home shift old policies new retreat responsibilities better way progressnnabroad home key new responsibilities l
ies placing division responsibility lived long consequences attempting gather power responsibility washingtonnnabroad home time
come turn away condescending policies paternalism washington knows bestnna person expected act responsibly responsibility human
nature let us encourage individuals home nations abroad decide let us locate responsibility places let us measure others themse
lvesnnthat today offer promise purely governmental solution every problem lived long false promise trusting much government ask
ed deliver leads inflated expectations reduced individual effort disappointment frustration erode confidence government people

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)

President Franklin D. Roosevelt in 1941:

```
nltk.FreqDist(all_Words).most_common(3)
```

```
[('nation', 10), ('know', 10), ('us', 8)]
```

President John F. Kennedy in 1961:

```
nltk.FreqDist(all_Words).most_common(3)
```

```
[('let', 11), ('us', 11), ('sides', 8)]
```

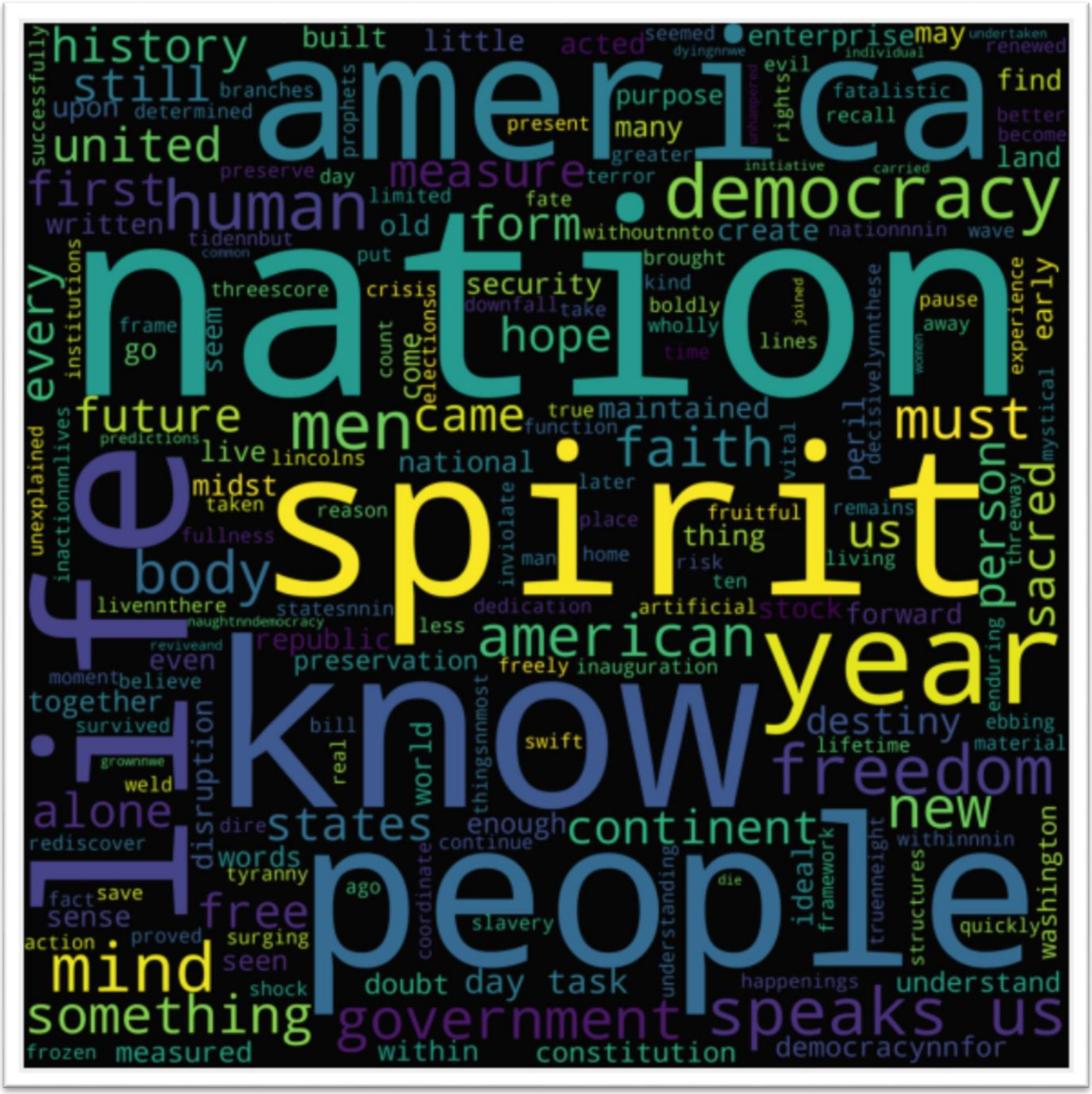
President Richard Nixon in 1973:

```
nltk.FreqDist(all_Words).most_common(3)
```

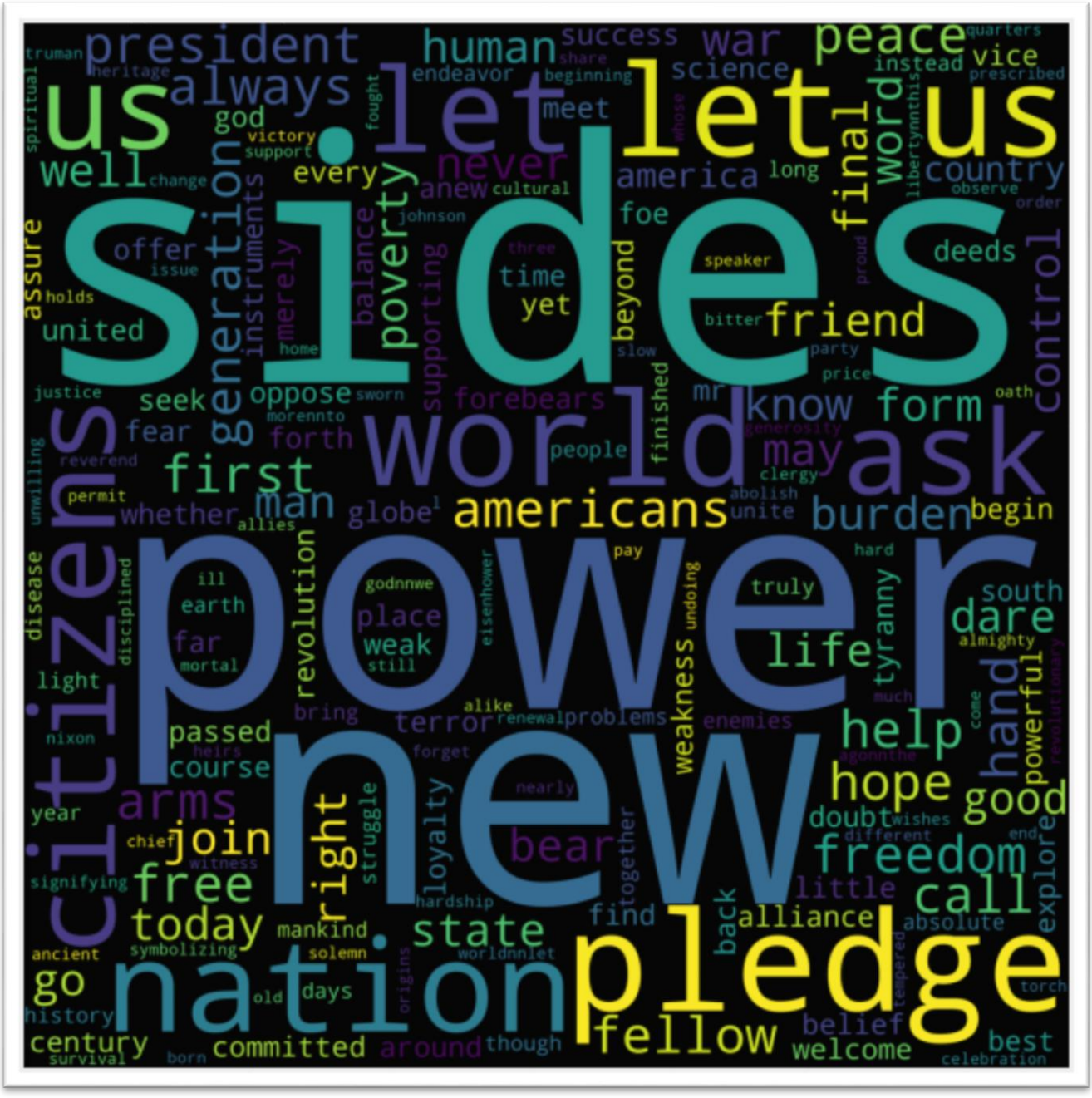
```
[('us', 26), ('new', 15), ('peace', 15)]
```

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

President Franklin D. Roosevelt in 1941:



President John F. Kennedy in 1961:



President Richard Nixon in 1973: