Machine learning

1/8/21

Mandira roy
Great learning

CONTENT

Problem 1:

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

Dataset for Problem: Election_Data.xlsx

Data Ingestion: 11 marks

1.1 Read the dataset. Do the descriptive statistics and do the null value condition check. Write an inference on it. (4 Marks)

1.2 Perform Univariate and Bivariate Analysis. Do exploratory data analysis. Check for Outliers. (7 Marks)

Data Preparation: 4 marks

1.3 Encode the data (having string values) for Modelling. Is Scaling necessary here or not? Data Split: Split the data into train and test (70:30). (4 Marks)

Modeling: 22 marks

1.4 Apply Logistic Regression and LDA (linear discriminant analysis). (4 marks)

1.5 Apply KNN Model and Naïve Bayes Model. Interpret the results. (4 marks)

1.6 Model Tuning, Bagging (Random Forest should be applied for Bagging), and Boosting. (7 marks)

1.7 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model. Final Model: Compare the models and write inference which model is best/optimized. (7 marks)

Inference: 5 marks

1.8 Based on these predictions, what are the insights? (5 marks)

ANSWER:

1.1. Before moving ahead with the exploration of data we will take a look at the data.

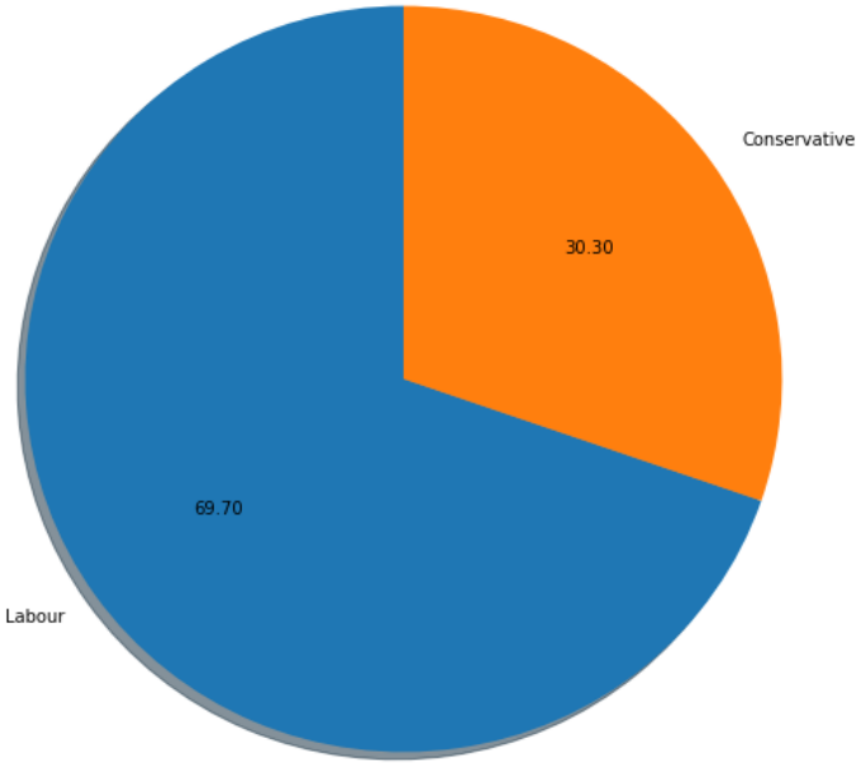
	Unnamed: 0	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	1	Labour	43	3	3	4	1	2	2	female
1	2	Labour	36	4	4	4	4	5	2	male
2	3	Labour	35	4	4	5	2	3	2	male
3	4	Labour	24	4	2	2	1	4	0	female
4	5	Labour	41	2	2	1	1	6	2	male
5	6	Labour	47	3	4	4	4	4	2	male
6	7	Labour	57	2	2	4	4	11	2	male
7	8	Labour	77	3	4	4	1	1	0	male
8	9	Labour	39	3	3	4	4	11	0	female
9	10	Labour	70	3	2	5	1	11	2	male
10	11	Labour	39	3	3	1	2	7	0	female

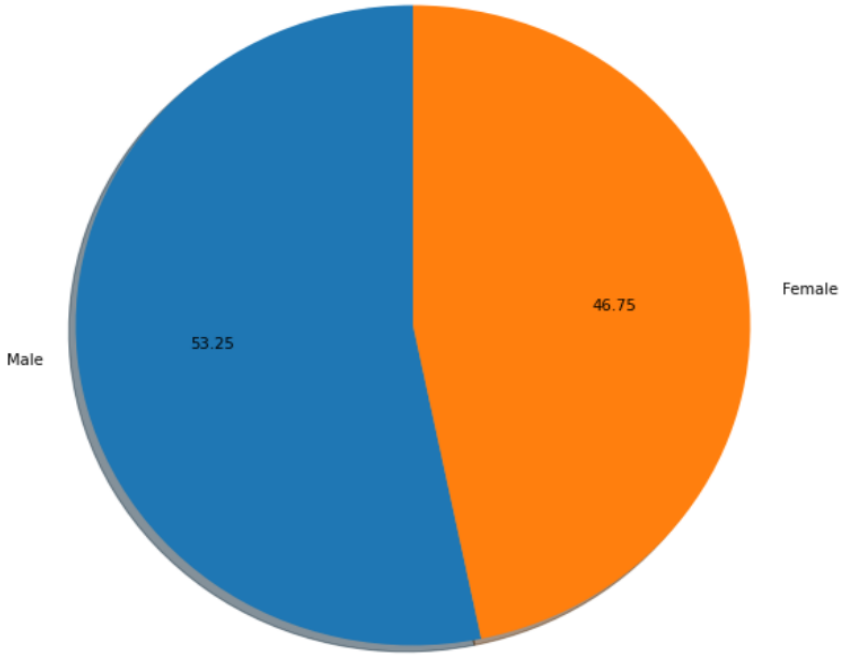
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   vote                    1525 non-null   object
 1   age                     1525 non-null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

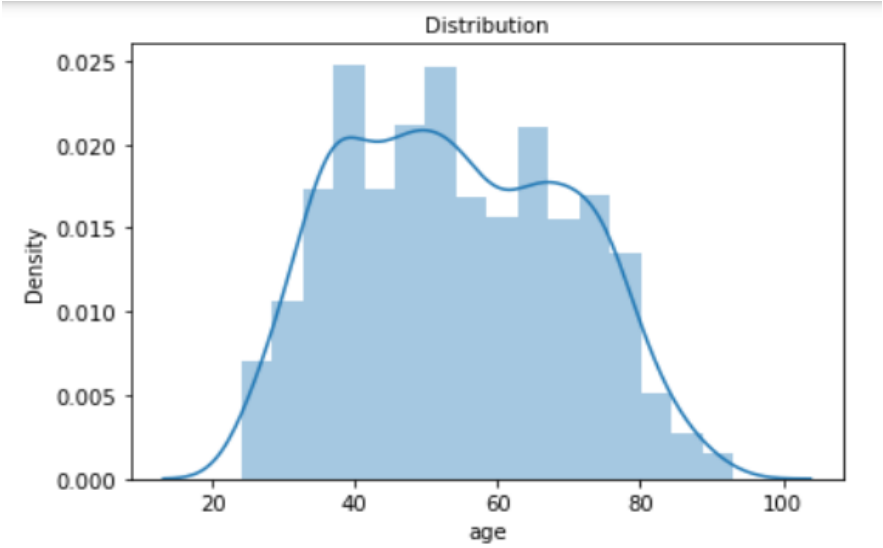
Inference: Here we can see that the data consist of 1525 rows and 10 columns but the Unnamed: 0 is removed as it was not needed in the further analysis and clustering , with no null and 8 duplicate values having int64 and object data type.

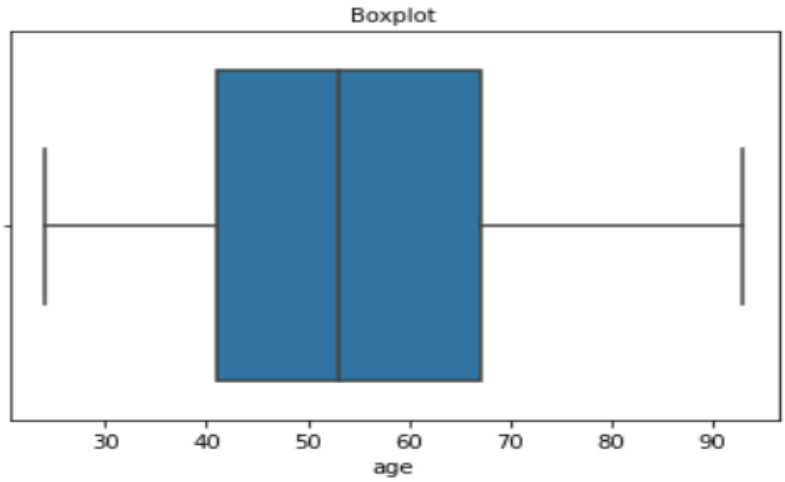
1.2:

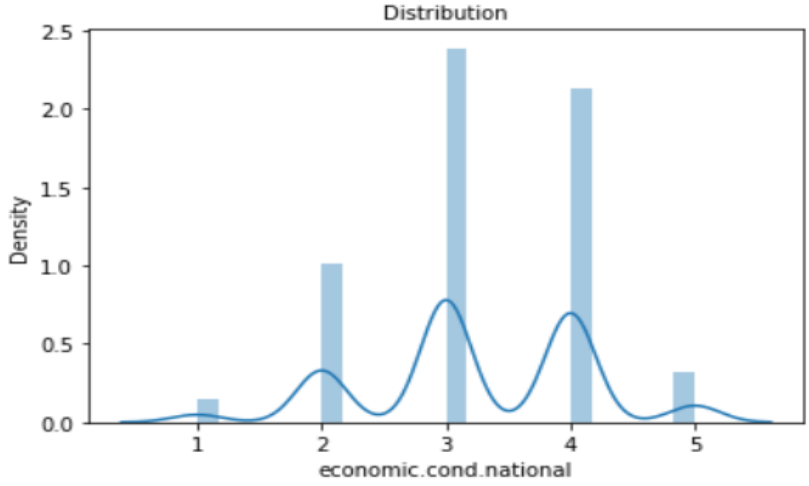
UNIVARIATE ANALYSIS:

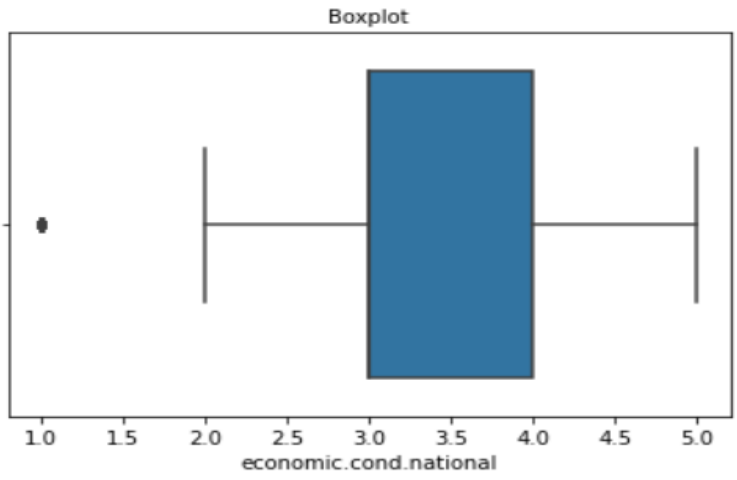


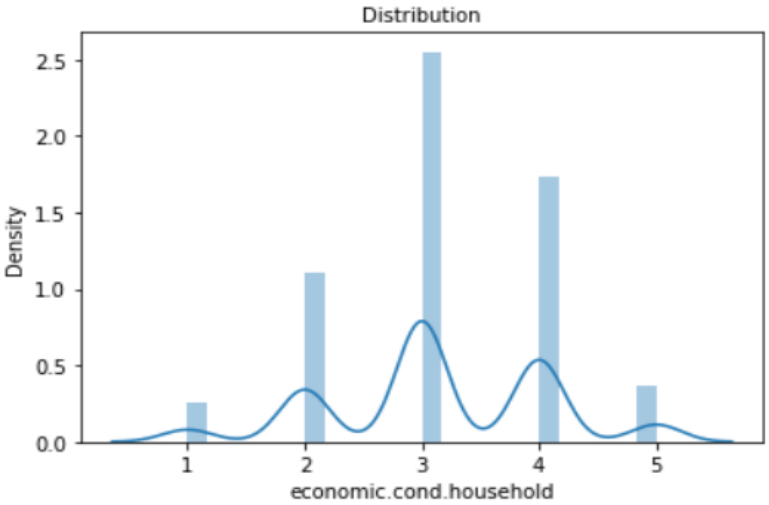

Male
53.25
46.75
Female

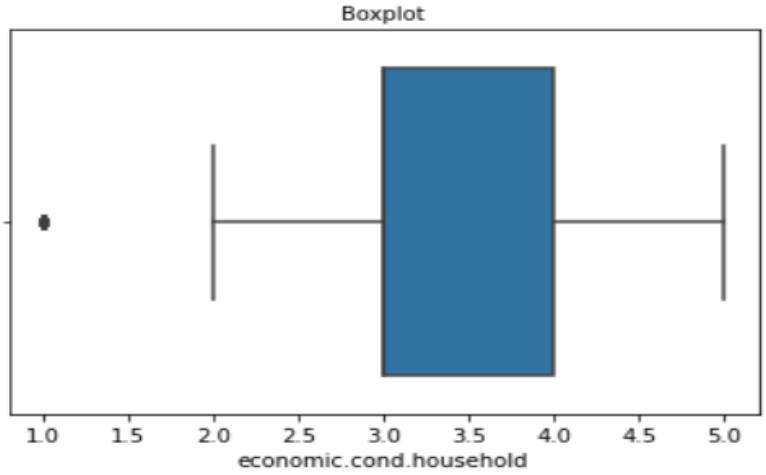

Distribution
Density
0.025
0.020
0.015
0.010
0.005
0.000
20
40
60
80
100
age

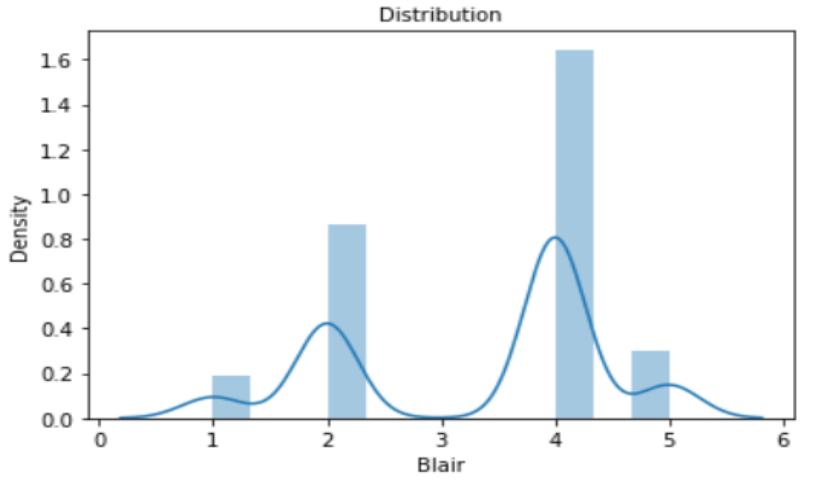

Boxplot
30
40
50
60
70
80
90
age

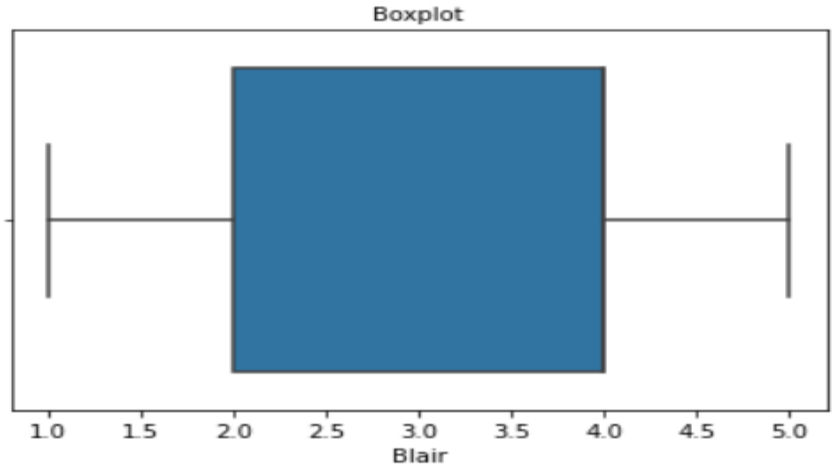

Distribution
2.5
2.0
1.5
1.0
0.5
0.0
Density
1
2
3
4
5
economic.cond.national

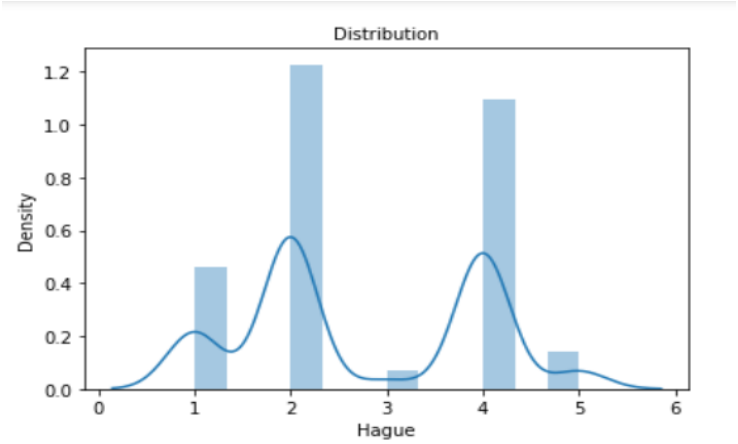

Boxplot
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
economic.cond.national

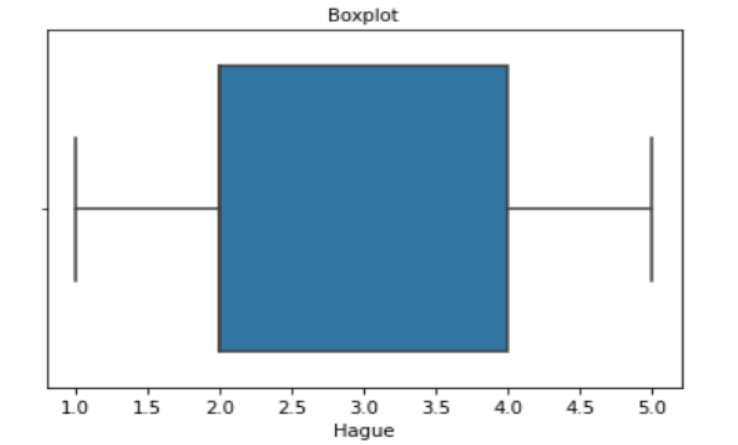

Distribution
Density
2.5
2.0
1.5
1.0
0.5
0.0
1
2
3
4
5
economic.cond.household

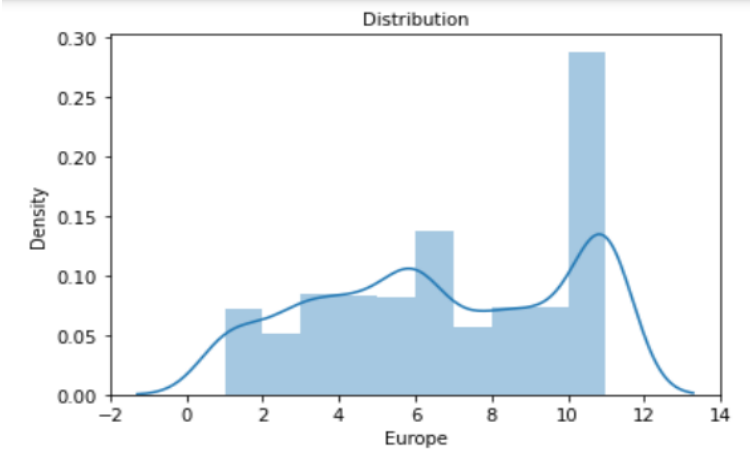

Boxplot
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
economic.cond.household

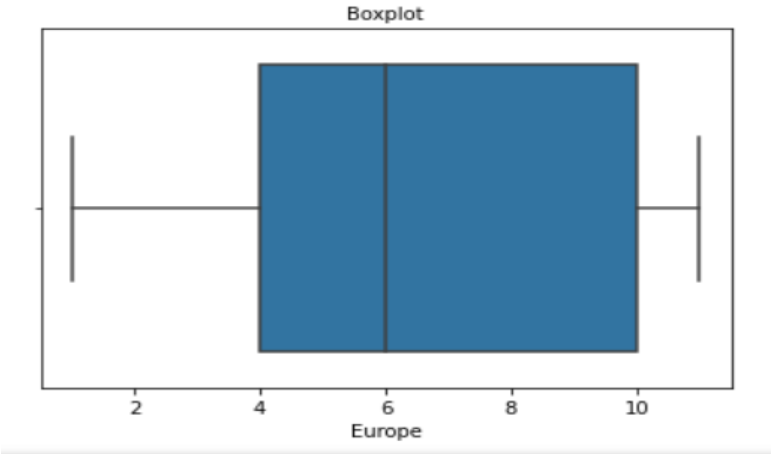

Distribution
Density
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
Blair

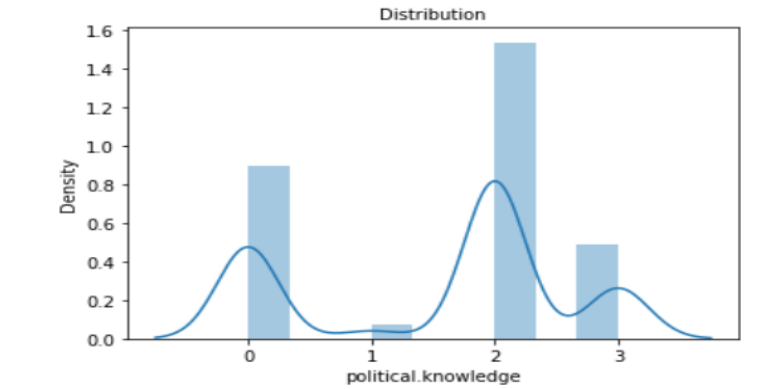

Boxplot
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Blair

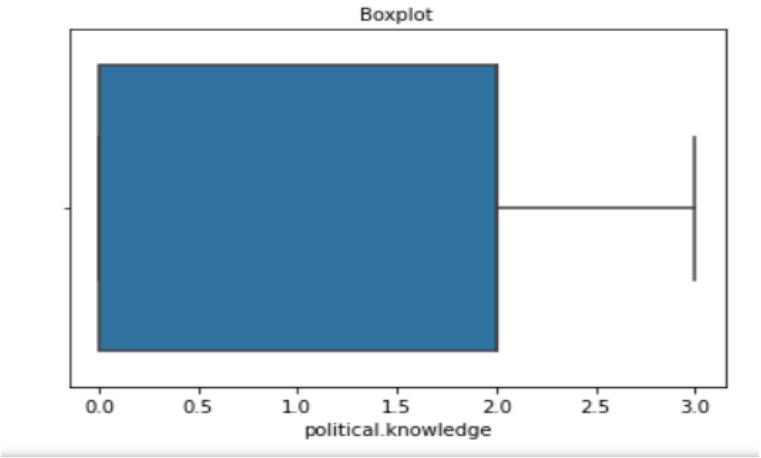

Distribution
Density
Hague
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
1
2
3
4
5
6


Boxplot
Hague
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Distribution
Density
Europe
0.00
0.05
0.10
0.15
0.20
0.25
0.30
−2
0
2
4
6
8
10
12
14



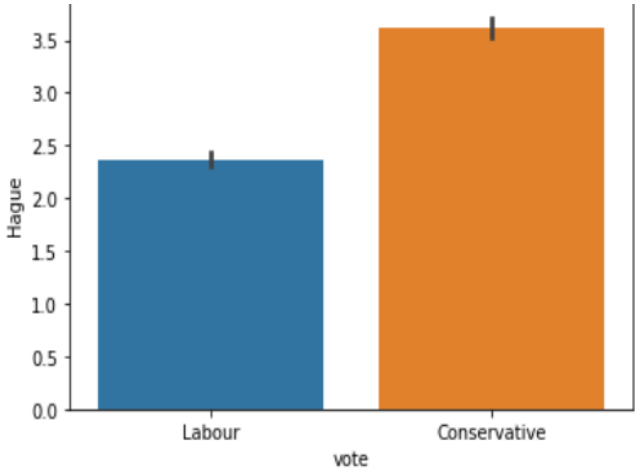


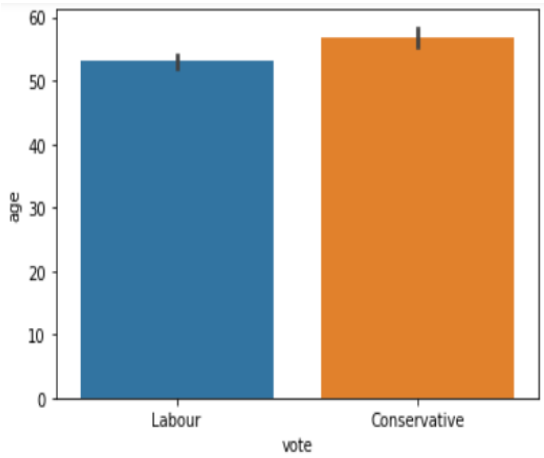


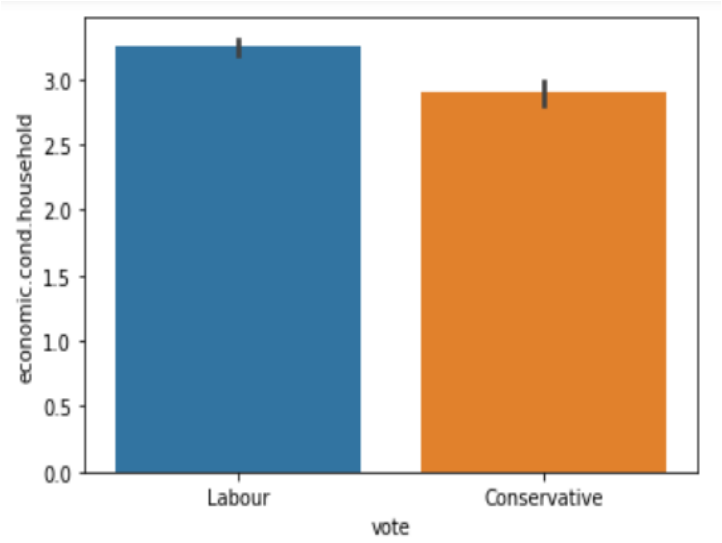
INFERENCE:

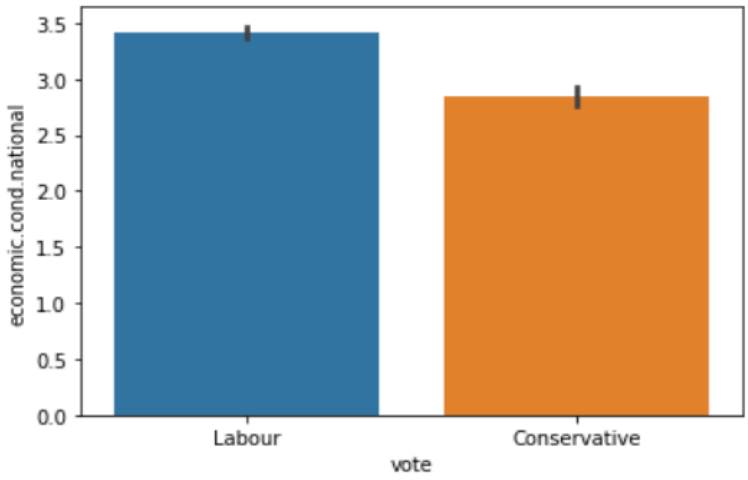
We can see from the above figure that the vote column is balanced in a 70:30 proportion and the gender column is balanced in a 54:46 proportion . There is no outlier, only one outlier in the economic.cond.household and economic.cond.national .

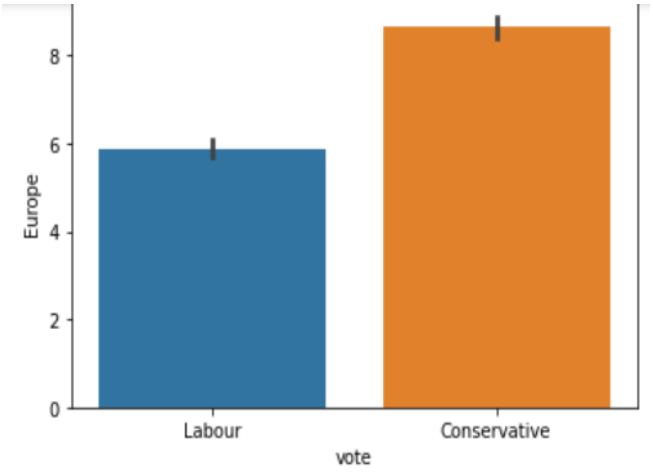
BIVARIATE AND MULTIVARIATE ANALYSIS:

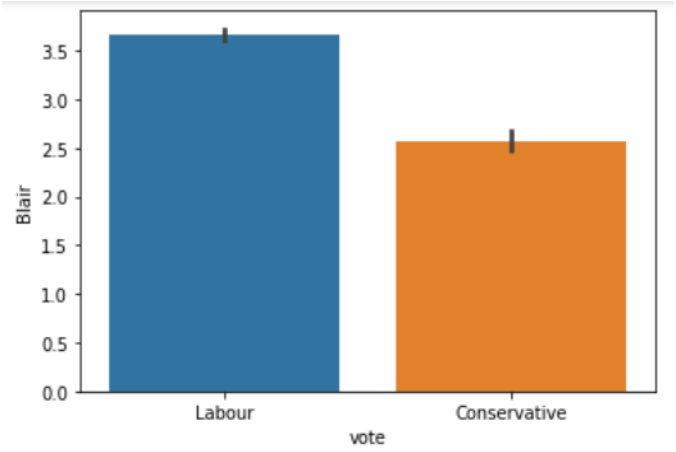


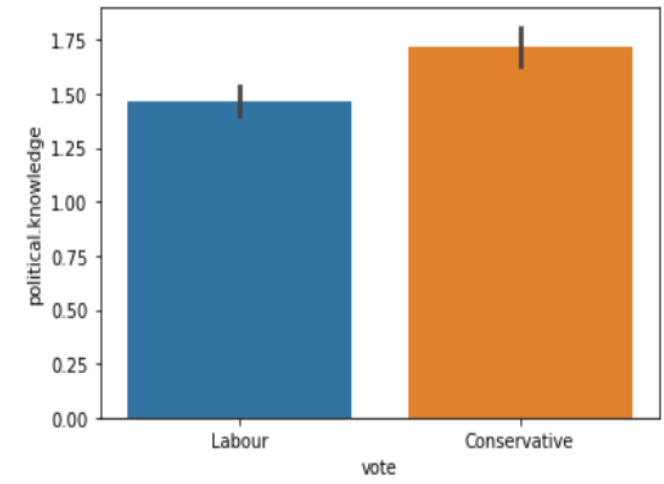






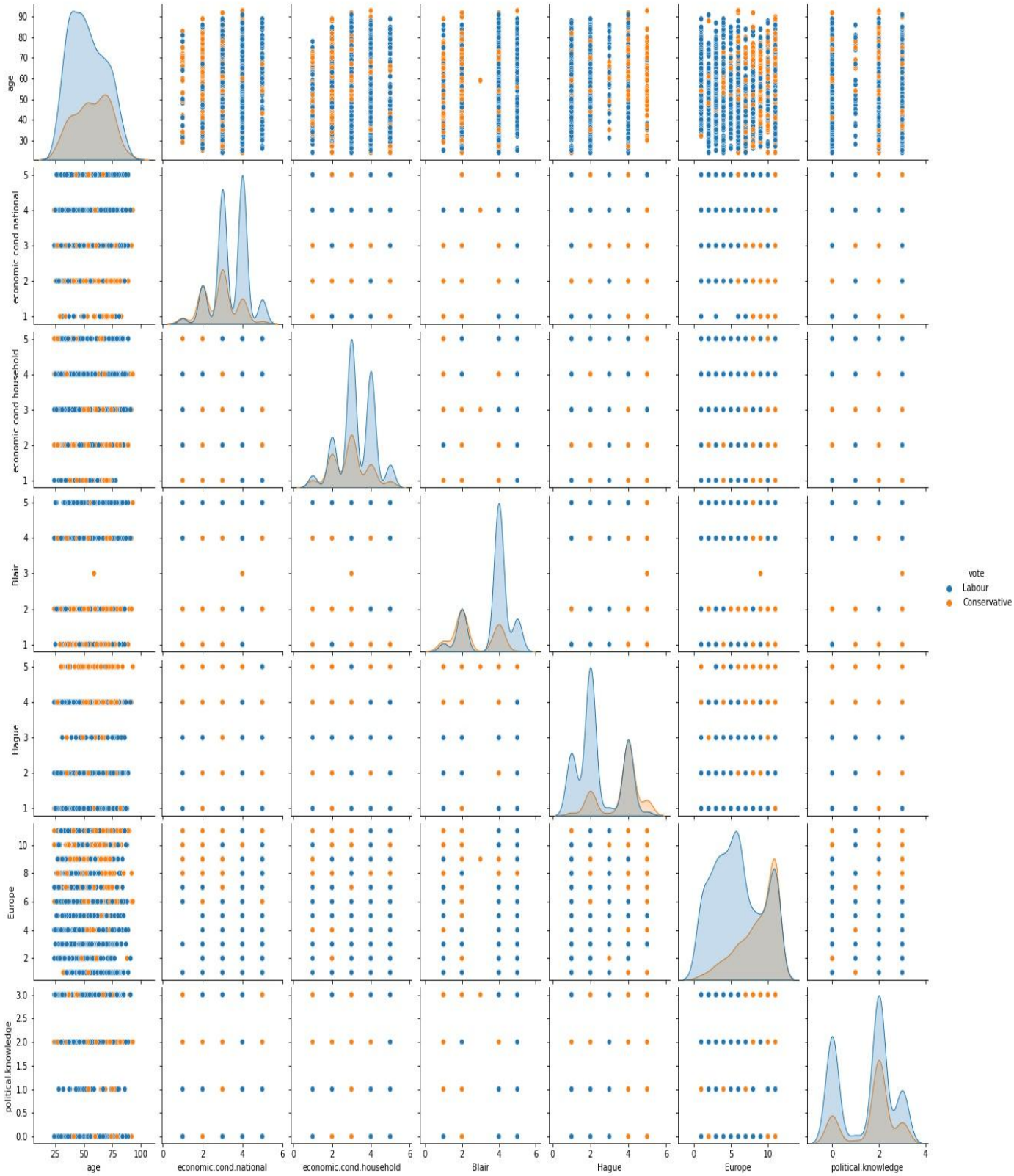


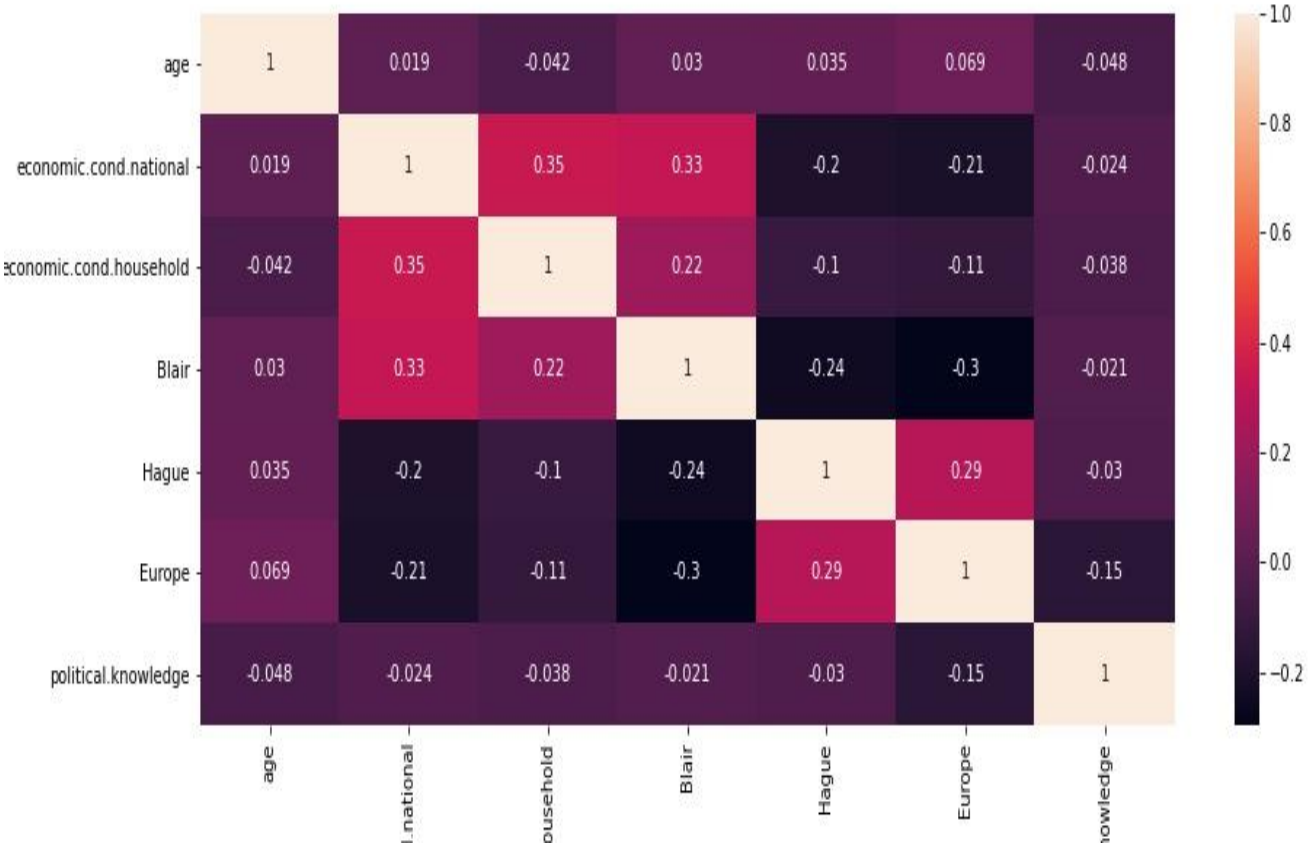




Inference: I have done the bivariate analysis of all the columns against the vote column to see how the other column is affecting the vote of labour and consecutive.

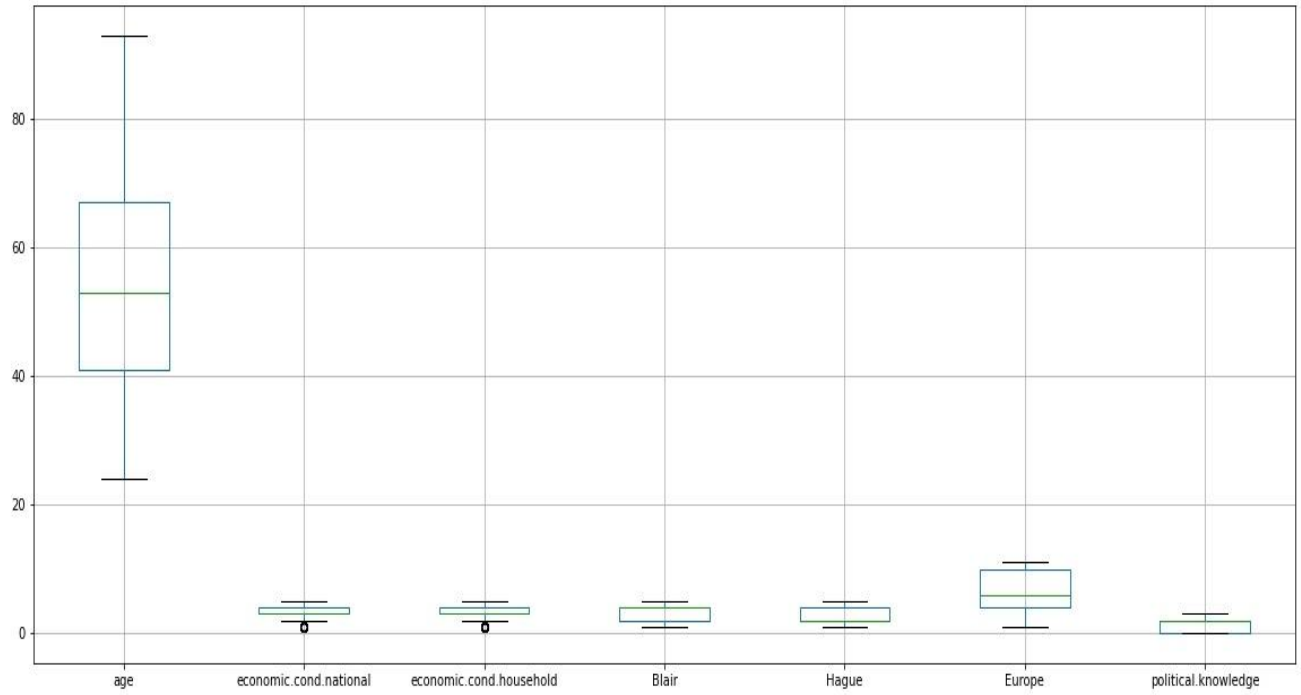
We can see in the political knowledge vs vote figure the consecutive votes are higher rather than the labour voters.


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



Inference:

As we can notice in the heat map the highest 3 correlation is between economic.cond.national vs economic.cond.household(0.35) , economic.cond.national vs blair(0.33),economic.cond.household vs blair(0.22). The top 3 negative correlations are between Hague vs blair(-0.24), economic.cond.national vs europe(-0.21), economic.cond.national vs Hague(-0.2).



Inference:

There is no outlier in the data but only two column i.e., economic.cond.national and economic.condi.household has one outlier each. There may be some voters having the economic condition relatively very low and facing an economic crisis .

1.3. Inference: The categorical data has been encoded like the vote and gender column.

Yes in this case the scaling is not necessary but is a good practice to perform as the mean,min,max value are't overlapping and there is a little difference in the scale of the data so here we may or may not do scaling as the difference is not that high among the data.As the ranges of some features are not overlapping we can perform the scaling which is necessary for some model building purposes.

The data has been divided into 70:30 proportions of train and test .

1.4. I have built the LDA and Logistic relation model.

Inference:For the logistic relation I have used these parameters max_iter=10000, n_jobs=2, penalty='none', solver='newton-cg',verbose=True. I have used this parameters as i have tried many values but this was giving me the best value so i have use these here.

For the LDA I haven't used any such techniques, I have done some normal model building .

The accuracy score of the LDA model train data is: 83.69%

The accuracy score of the LDA model test data is: 81.87%

The accuracy score of the Logistic model train data is: 84.06%

The accuracy score of the Logistic model test data is: 82.31%

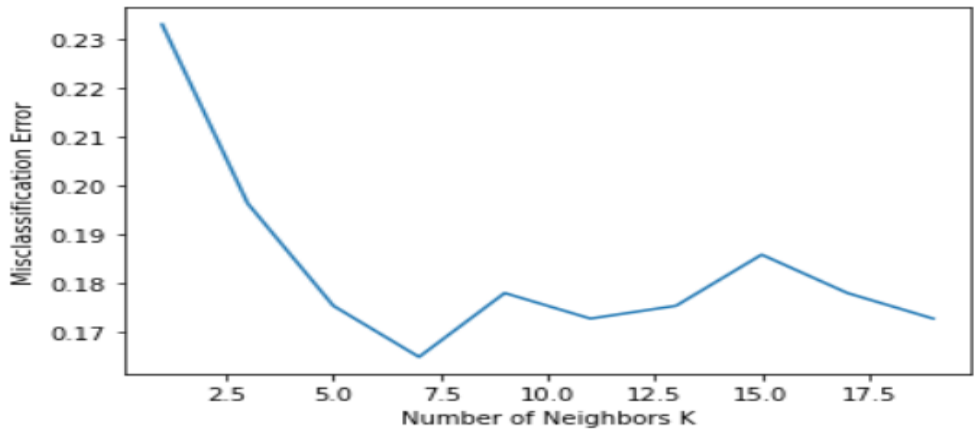
The difference in the test and train data accuracy is not greater than 10% so we can't say that it's not a overfitting model though the accuracy of train data is slightly higher than the test data in both the cases .

1.5. I have built the naive bayes model and the knn model .

Inference: I haven't done any analysis for the naive bayes model building,I have simply created a model. For the knn model I have tried the k values and as I have noticed there is no further drastic change after the k value 5 .

```
[0.2329842931937727,
 0.19633507853403143,
 0.17539267015706805,
 0.16492146596858637,
 0.17801047120418845,
 0.17277486910994766,
 0.17539267015706805,
 0.18586387434554974,
 0.17801047120418845,
 0.17277486910994766]
```

So I have build the knn model on the k value 5.The MCE graph values are interpreting the same.



The accuracy score of the naive bayes model train data is: 83.31%

The accuracy score of the naive bayes model test data is: 82.53%

The accuracy score of the knn model train data is: 83.789%

The accuracy score of the knn model test data is: 82.46%

So we can clearly see that both models are really moring very good the accuracy score difference is not so high between the train and test set and in the knn model i have tried some k values but the k=5 gave the best accuracy score and the predictions for the test and train are real close to each other. So by this we can conclude that the models are predicting the values with accuracy close to 84% both for train and test data.And lastly the models are neither overfitting nor underfitting.

1.6 I have tuned the models and applied the grid search on each model .
Inference: 'criterion': 'gini', 'max_depth': 4, 'min_samples_leaf': 10, 'min_samples_split': 60

These were the best parameters suggested by the grid search and thus I used them to build the other models and do the analysis on them. I haven't applied the grid search on the above models as it was not mentioned for the above questions .

I have done the modeling in the python file and in the param_grid i have tried and tested different values which were giving better results according to those values i have deleted the values of parameters or i have used the default technique taught in one of the videos to use 0.1/0.2/0.3 part of the total value in min sample leaf and three times of min sample leaf is the min sample split.

	important
Hague	0.446956
Europe	0.209833
Blair	0.143883
political.knowledge	0.127099
age	0.046109
economic.cond.household	0.013970
economic.cond.national	0.012150
Male	0.000000

According to the feature importance table the Hague column has the max feature importance with the vote columns and in the above bar plot we have seen that conservation voters are more than the labour voter for the Hague data.

The accuracy score of the bagging model train data is: 83.37%

The accuracy score of the bagging model test data is: 80.56%

The accuracy score of the Ada boosting model train data is: 99.90%

The accuracy score of the Ada boosting model test data is: 78.38%

The accuracy score of the Gradient boosting model train data is: 87.81%

The accuracy score of the Gradient boosting model test data is: 83.84%

As we can see that the Ada boosting model we can see that the model isn't working well. The model is overfitting the train and test data set accuracy is varying more than 10% and it isn't right . Whereas others are working fine and they are not that overfitting the test training accuracy varies 3-4% which is fine to neglect . Otherwise the models are performing fine other than the Ada boosting.

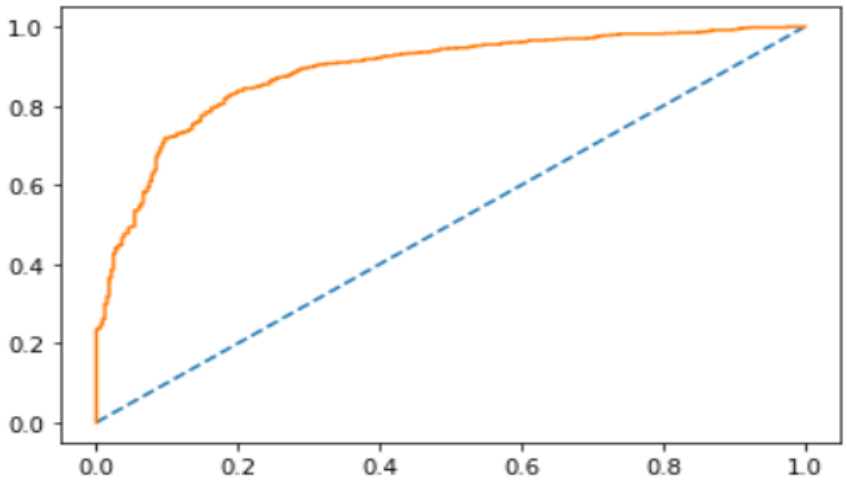
1.7.

NAIVE BAYES

TRAIN-

```
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
```

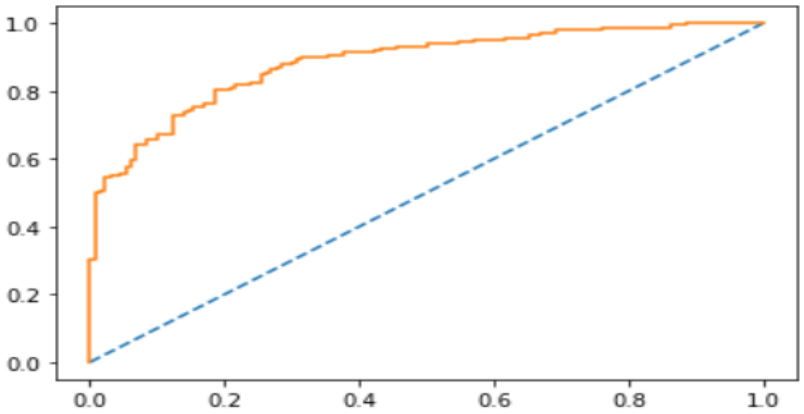
```
the auc 0.886
```



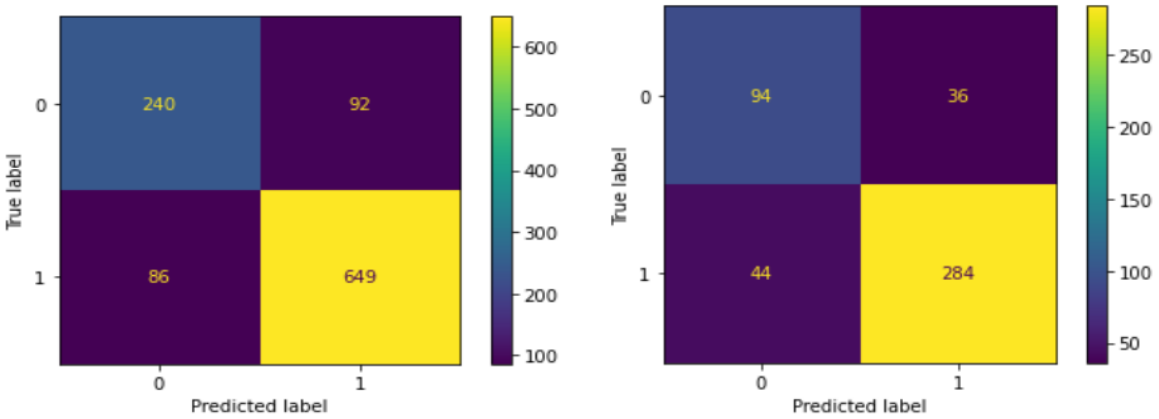
TEST-

```
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
```

```
the auc curve 0.885
[<matplotlib.lines.Line2D at 0x201664c1ee0>]
```



CONFUSION MATRIX:TRAIN AND TEST



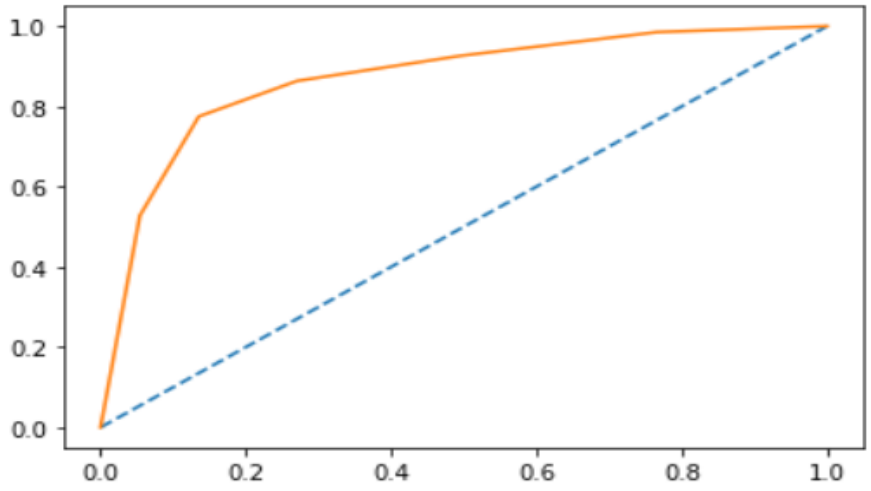
KNN

TRAIN

```
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
```

```
the auc curve 0.904
```

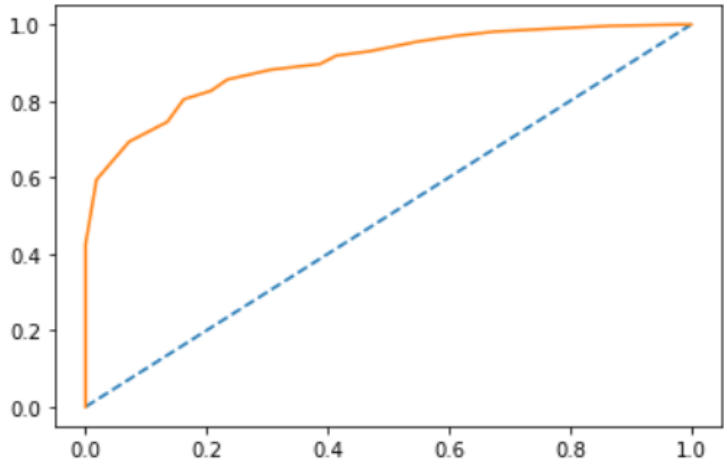
```
[<matplotlib.lines.Line2D at 0x20167176f10>]
```



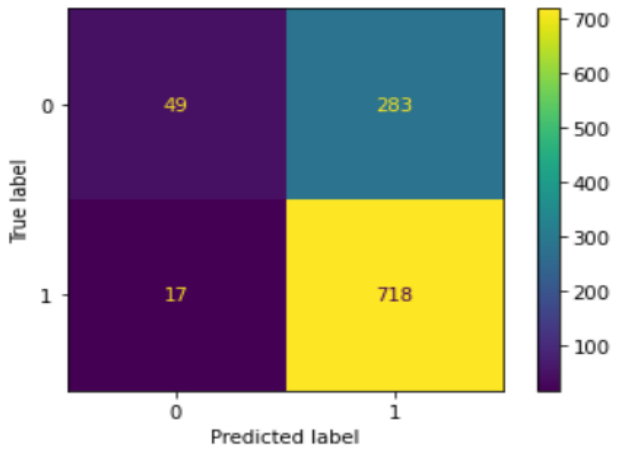
TEST

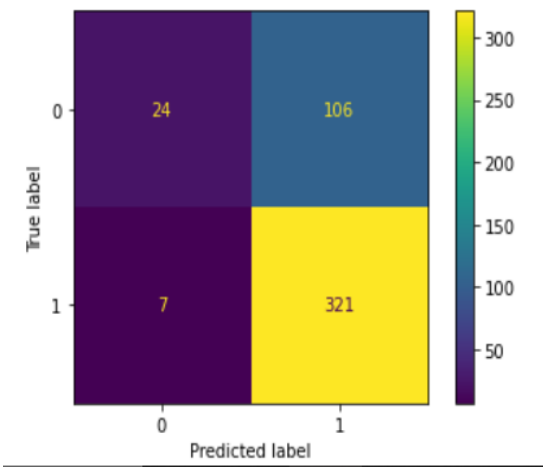
```
0.824607329842932
[[ 81  30]
 [ 37 234]]
              precision    recall  f1-score   support

           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
```

```
the auc curve 0.900

[<matplotlib.lines.Line2D at 0x201671d74c0>]
```



CONFUSION MATRIX:TRAIN AND TEST



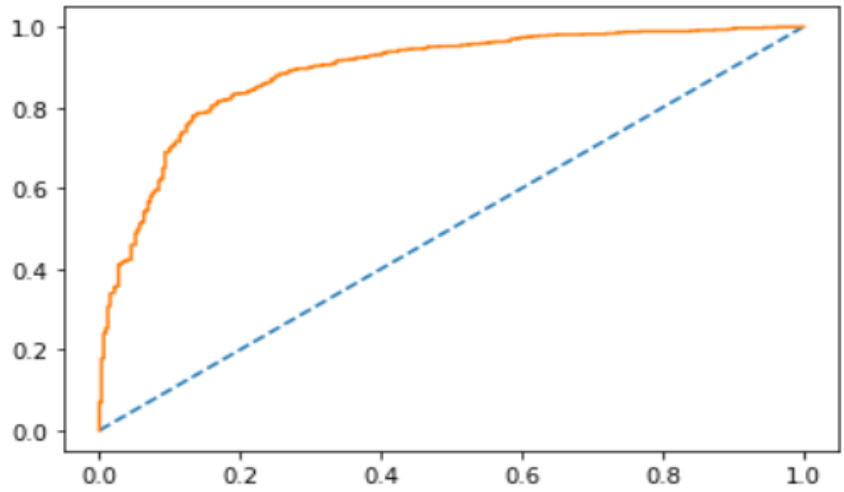


LDA:

TRAIN

```
0.8369259606373008
[[233  99]
 [ 75 660]]
              precision    recall  f1-score   support

           0       0.76      0.70      0.73       332
           1       0.87      0.90      0.88       735

    accuracy                           0.84      1067
   macro avg       0.81      0.80      0.81      1067
weighted avg       0.83      0.84      0.84      1067
```

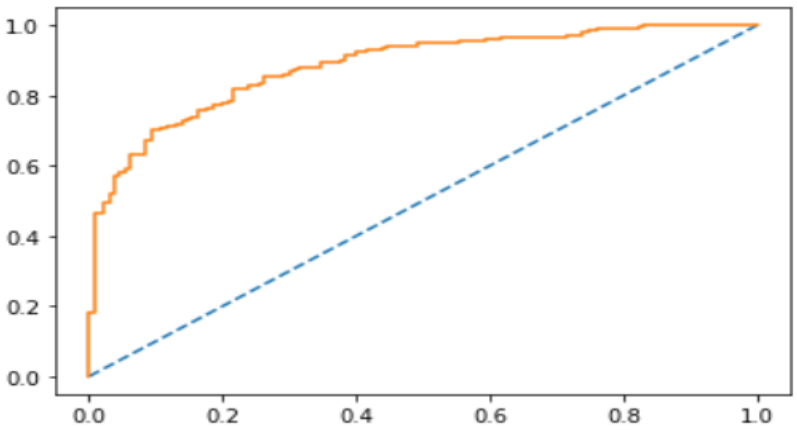
```
the auc 0.889
```



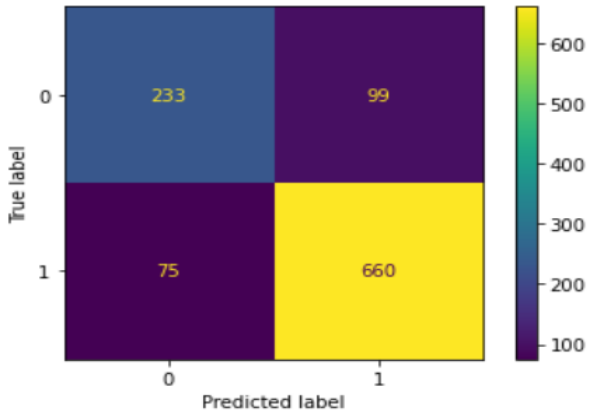
TEST

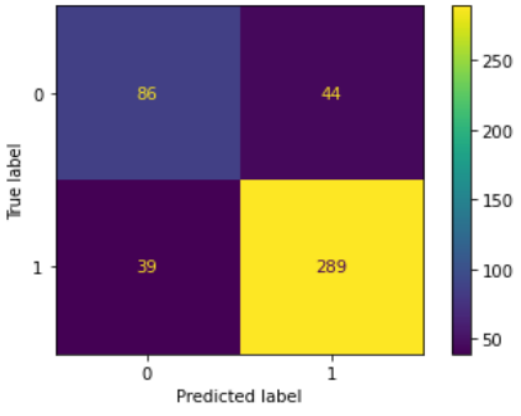
```
0.8187772925764192
[[ 86  44]
 [ 39 289]]
              precision    recall  f1-score   support

           0       0.69      0.66      0.67       130
           1       0.87      0.88      0.87       328

    accuracy                           0.82       458
   macro avg       0.78      0.77      0.77       458
weighted avg       0.82      0.82      0.82       458
```

```
the auc curve 0.884

[<matplotlib.lines.Line2D at 0x20167379640>]
```



CONFUSION MATRIX:TRAIN AND TEST



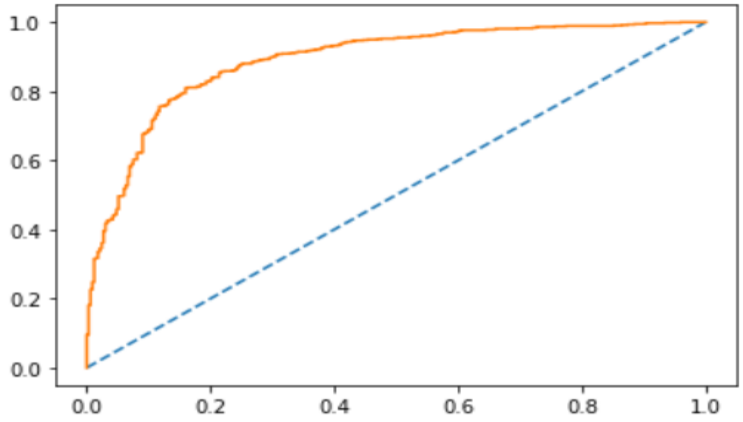


LOGISTIC REGRESSION:

TRAIN

```
0.8406747891283973
[[230 102]
 [ 68 667]]
              precision    recall  f1-score   support

           0       0.77      0.69      0.73       332
           1       0.87      0.91      0.89       735

    accuracy                           0.84      1067
   macro avg       0.82      0.80      0.81      1067
weighted avg       0.84      0.84      0.84      1067
```

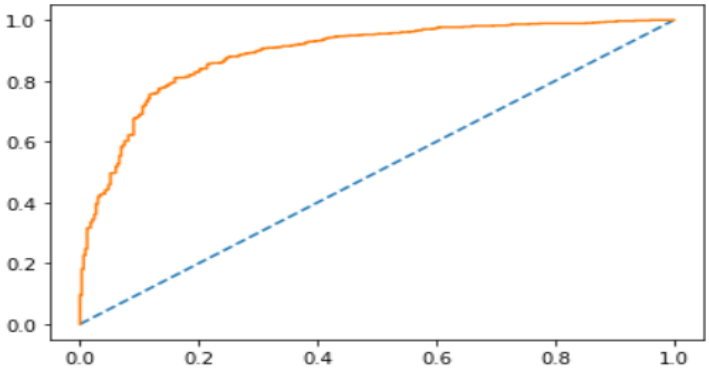
```
AUC: 0.889
```



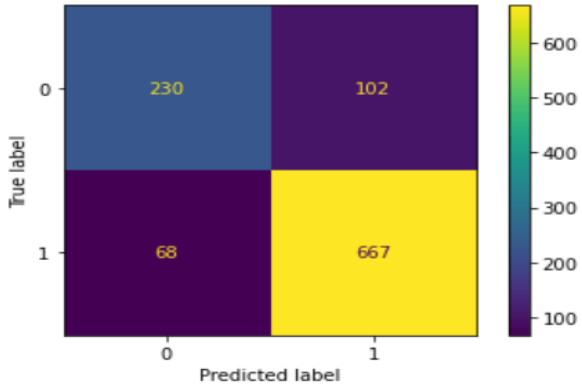
TEST

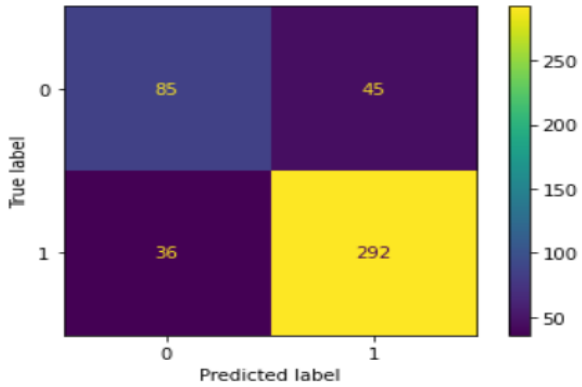
```
0.8231441048034934
[[ 85  45]
 [ 36 292]]
              precision    recall  f1-score   support

           0       0.70      0.65      0.68       130
           1       0.87      0.89      0.88       328

    accuracy                           0.82       458
   macro avg       0.78      0.77      0.78       458
weighted avg       0.82      0.82      0.82       458
```

AUC : 0.882



CONFUSION MATRIX:TRAIN TEST-





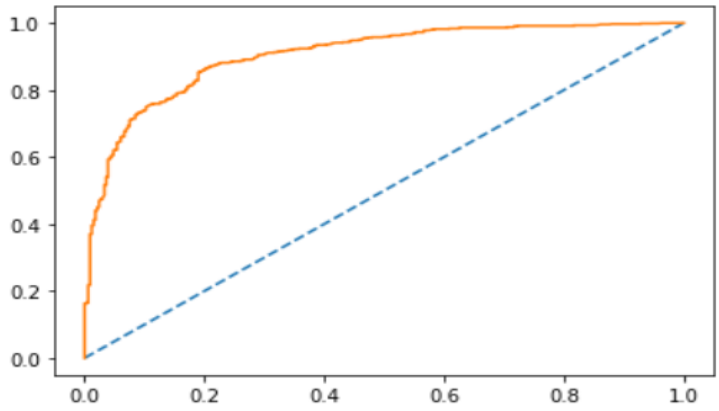
BAGGING-

TRAIN

```
0.837863167760075
[[217 115]
 [ 58 677]]
              precision    recall  f1-score   support

           0       0.79      0.65      0.71       332
           1       0.85      0.92      0.89       735

    accuracy                           0.84      1067
   macro avg       0.82      0.79      0.80      1067
weighted avg       0.83      0.84      0.83      1067
```

```
AUC: 0.905
```

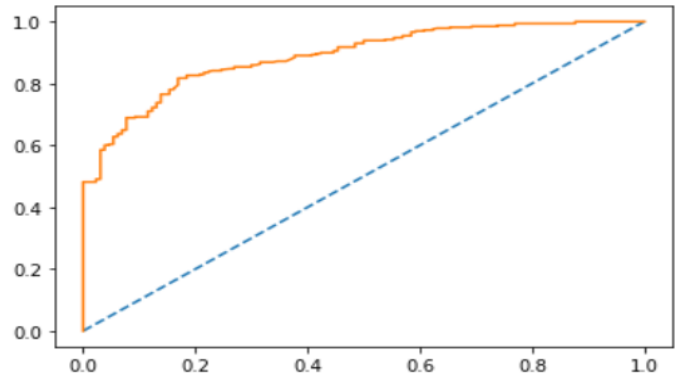
```
[<matplotlib.lines.Line2D at 0x20163f84a90>]
```



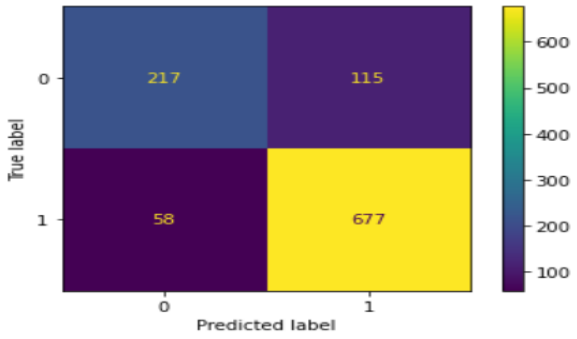
TEST

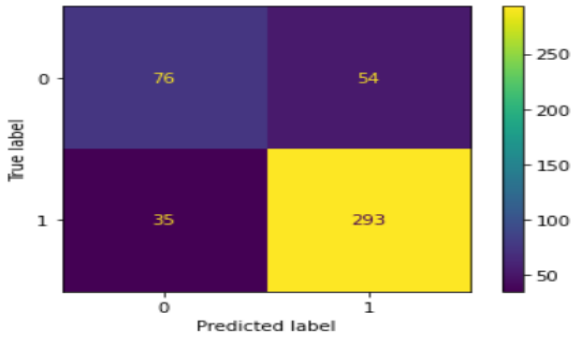
```
0.8056768558951966
[[ 76  54]
 [ 35 293]]
              precision    recall  f1-score   support

           0       0.68      0.58      0.63       130
           1       0.84      0.89      0.87       328

    accuracy                           0.81       458
   macro avg       0.76      0.74      0.75       458
weighted avg       0.80      0.81      0.80       458
```

AUC: 0.889



CONFUSION MATRIX: TRAIN AND TEST

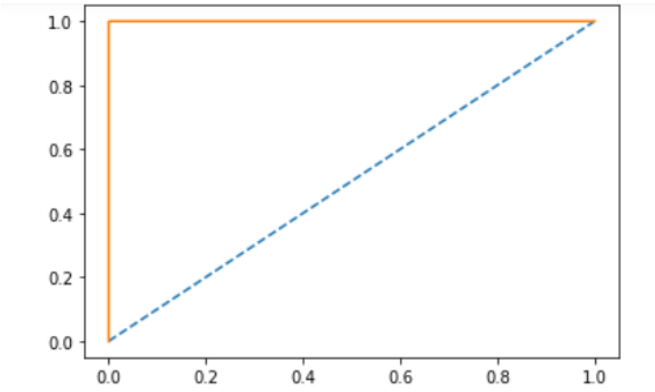




ADA BOOSTING

Train

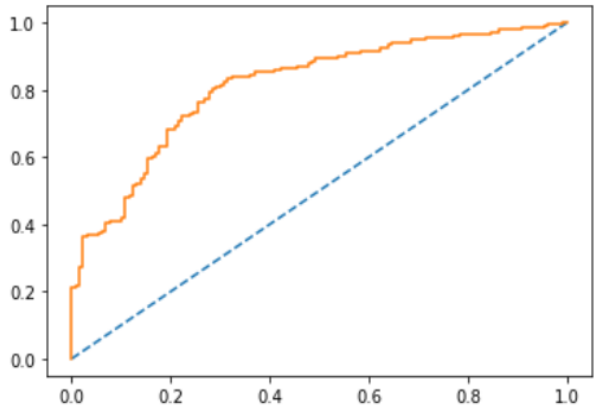
```
0.9990627928772259
[[331   1]
 [  0 735]]
              precision    recall  f1-score   support

           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
```



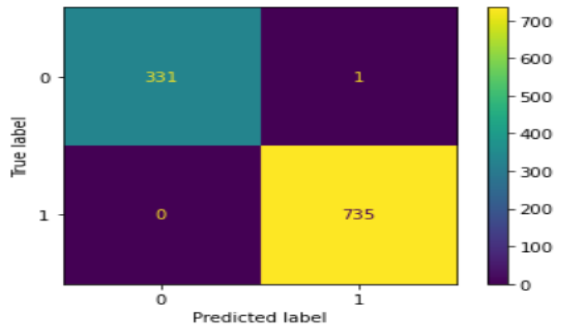
Test

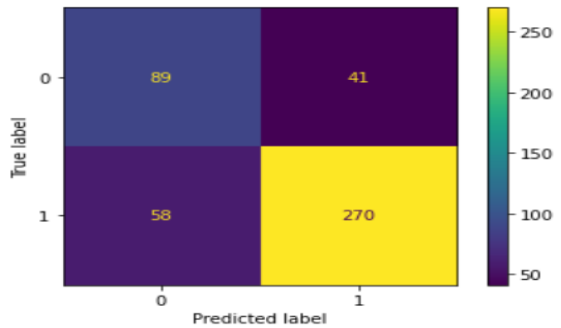
```
0.7838427947598253
[[ 94  36]
 [ 44 284]]
              precision    recall  f1-score   support

           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: 0.812
```



CONFUSION MATRIX: TRAIN AN TEST

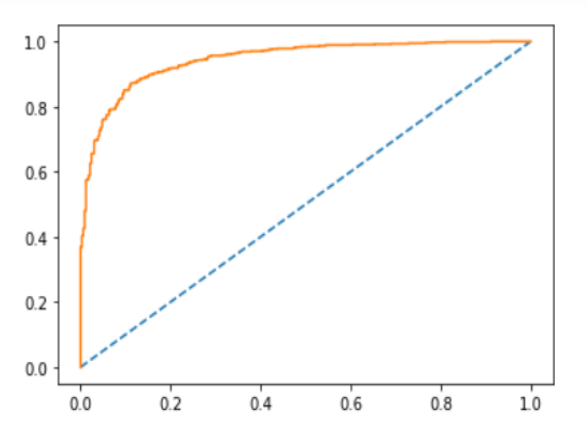




GRADIENT BOOSTING

TRAIN

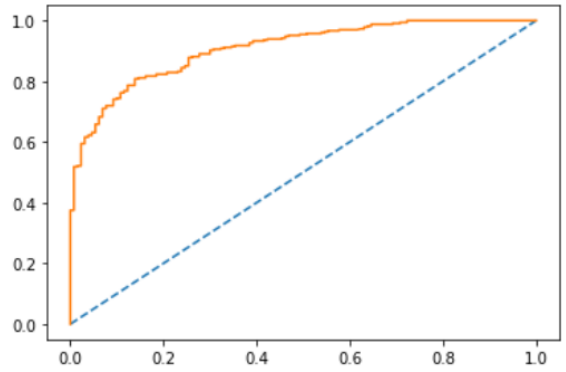
```
0.8781630740393627
[[240  92]
 [ 86 649]]
              precision    recall  f1-score   support

           0       0.83      0.77      0.80       332
           1       0.90      0.93      0.91       735

    accuracy                           0.88      1067
   macro avg       0.86      0.85      0.85      1067
weighted avg       0.88      0.88      0.88      1067
```



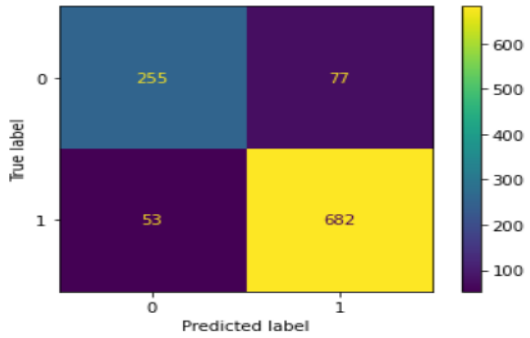
TEST

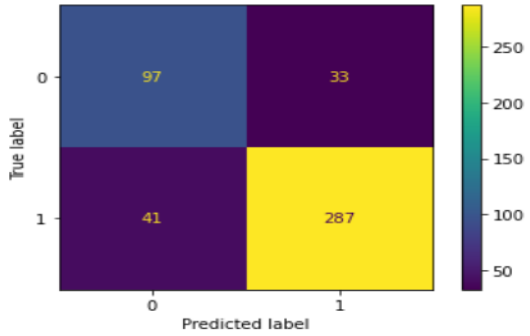
```
0.8384279475982532
[[ 94  36]
 [ 44 284]]
              precision    recall  f1-score   support

           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: 0.908

[<matplotlib.lines.Line2D at 0x201630f48e0>]
```



CONFUSION MATRIX: TRAIN TEST:





From the above analysis an model we can say that the naive bayes an knn models are showing the best performance among all the models .

1.8: For the above problem we need more data for the better prediction .But the data in hand is also showing quite good results for the votes .If the channel wants to predict for better prediction the channel must cover the Hague as if it will be a good predictor feature. The channel should cover more economic.cond.household and economic.cond.national part so as the data there is correlated and in real case scenario the economic status affects the elections very much.The conservative voters are more than the labour voters are effecting the data focus on that may help the channel to grow.