Machine Learning Project

Balasubramaniyam Ravichandran

11/13/22

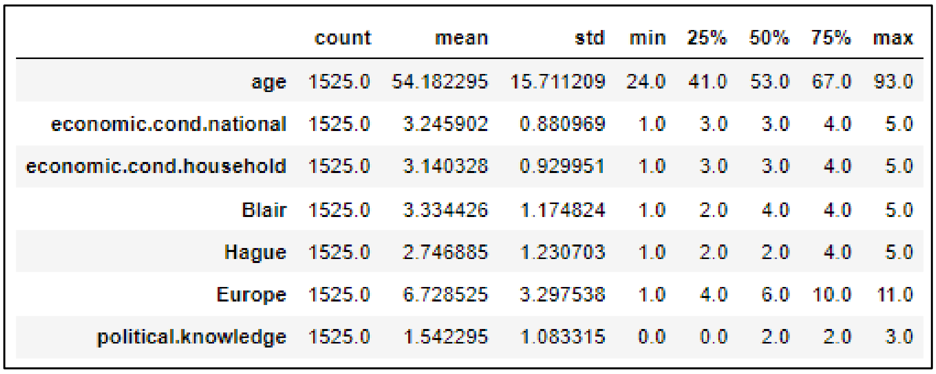
INDEX

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.

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.**

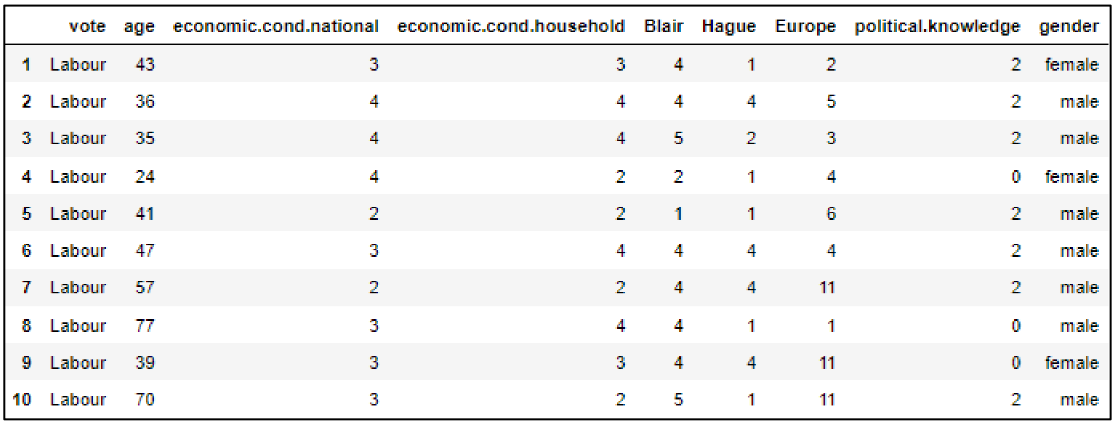
The dataset set is read, and suitable descriptive summary is made as follows,



	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

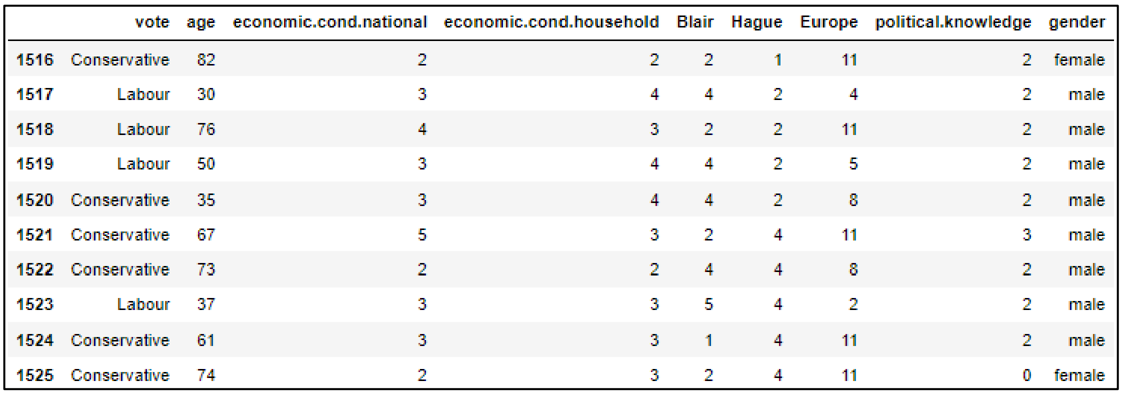
1.1.1. Data summary

There are some features like the economic.cond.national, economic.cond.household, Blair, Hague, Europe, political.knowledge in our dataset which are of ordinal nature. These features are a measure of opinion or the stand of voters, parties, leader and are rated from 1 up to 5, 11. ‘1’ being bad or worst and ‘5’,’11’ being good or excellent. The age feature is continuous in nature.



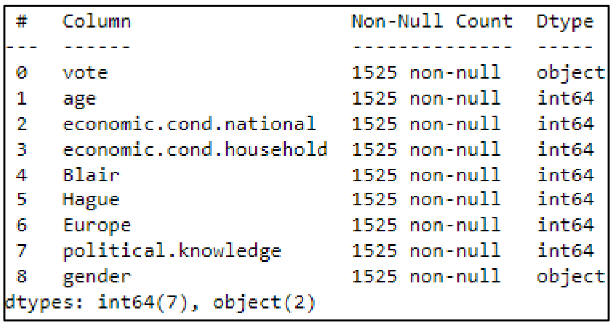
	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
1	Labour	43	3	3	4	1	2	2	female
2	Labour	36	4	4	4	4	5	2	male
3	Labour	35	4	4	5	2	3	2	male
4	Labour	24	4	2	2	1	4	0	female
5	Labour	41	2	2	1	1	6	2	male
6	Labour	47	3	4	4	4	4	2	male
7	Labour	57	2	2	4	4	11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

1.1.2. First ten entries of the dataset.



	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
1516	Conservative	82	2	2	2	1	11	2	female
1517	Labour	30	3	4	4	2	4	2	male
1518	Labour	76	4	3	2	2	11	2	male
1519	Labour	50	3	4	4	2	5	2	male
1520	Conservative	35	3	4	4	2	8	2	male
1521	Conservative	67	5	3	2	4	11	3	male
1522	Conservative	73	2	2	4	4	8	2	male
1523	Labour	37	3	3	5	4	2	2	male
1524	Conservative	61	3	3	1	4	11	2	male
1525	Conservative	74	2	3	2	4	11	0	female

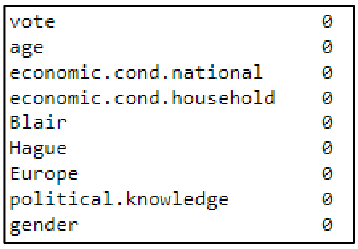
1.1.3. Last ten entries of the dataset.



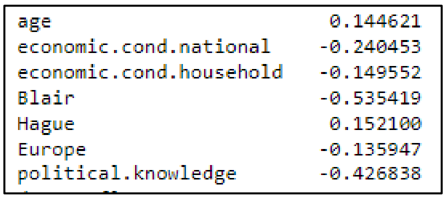
```
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   vote                     1525 non-null   object
 1   age                      1525 non-null   int64
 2   economic.cond.national   1525 non-null   int64
 3   economic.cond.household  1525 non-null   int64
 4   Blair                    1525 non-null   int64
 5   Hague                    1525 non-null   int64
 6   Europe                   1525 non-null   int64
 7   political.knowledge      1525 non-null   int64
 8   gender                   1525 non-null   object
dtypes: int64(7), object(2)
```

1.1.4. Information of data

The dataset is having a total of 9 columns, 1525 rows. The Unnamed:0 column was dropped as it was not useful in model building. The dataset has 2 columns (vote, gender) of object data type and the remaining 7 columns are of integer datatype.



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



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

1.1.5. Null check and skewness check

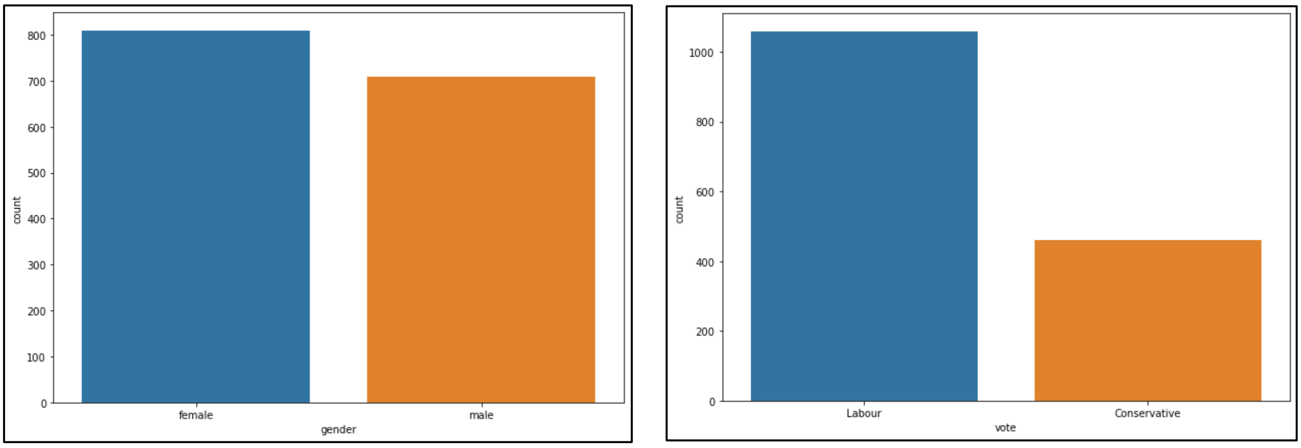
The dataset is checked for null values and observed that there are no null values in the dataset. The dataset is checked for skewness, If the skewness is between -0.5 and 0.5, the data is symmetrical, If the skewness is between -1 and – 0.5 or between 0.5 and 1, the data is moderately skewed. If the skewness is less than -1 or greater than 1, the data is highly skewed. Based on this rule it is observed that the feature ‘Blair’ is slightly skewed, and all other features are distributed symmetrically.

1.2) **Perform EDA (Check the null values, Data types, shape, Univariate, bivariate analysis). Also check for outliers. Interpret the inferences for each. 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.**

The null value check, data type checks are already discussed in previous question. The shape of the dataset is (Rows = 1525, Columns = 9). 8 duplicate rows were found and are dropped from the dataset.

Univariate Analysis,

The dataset is having two categorical data type features. Gender and the Vote features.



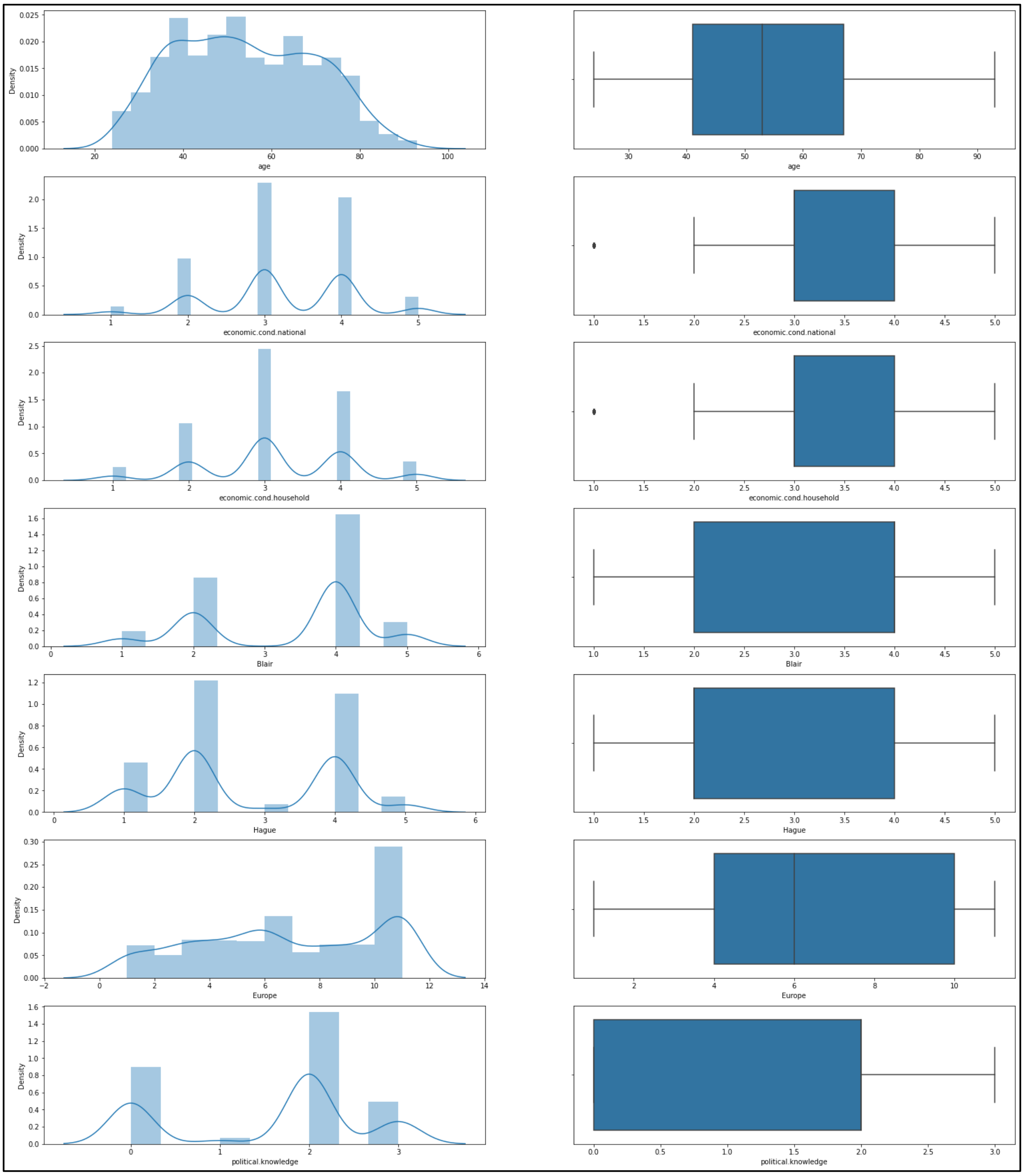
1.2.1. Gender and Vote (Categorical features)

- There are 808 female voters (53.3%), 709 male voters (46.7%) in our dataset.
- Party choice of 1057 voters is Labour(69.68%), 460 voters are Conservative(30.32%).

The univariate analysis of continuous variable

From the plots of the continuous variables, it is observed that the datapoints of these features are multi modal in nature. All the continuous variables are having more than one mode.

The boxplot of the continuous features shows some outliers present in the economic.cond.national, economic.cond.household. There is no need to treat these outliers as they are an indication of the voter’s opinion on the economic conditions and within an acceptable data range (1 to 5). The min, max, mean values of the boxplot are not clear in some of the variables, these values need to calculated separately if needed further down the process.



1.2.2. Histogram and Boxplots

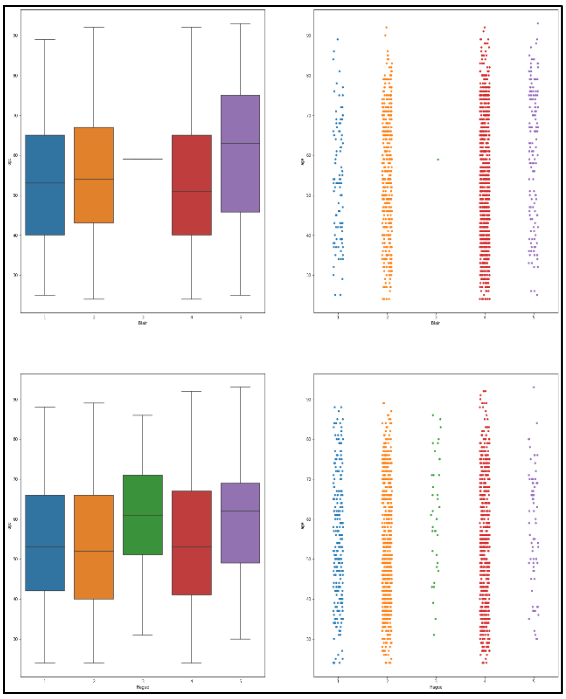
(Continuous features)

Bivariate analysis

Bivariate analysis helps in understanding the relationship between the variables of out dataset. The pair plot and heatmap helps in understanding the correlation between the continuous variables of our dataset.

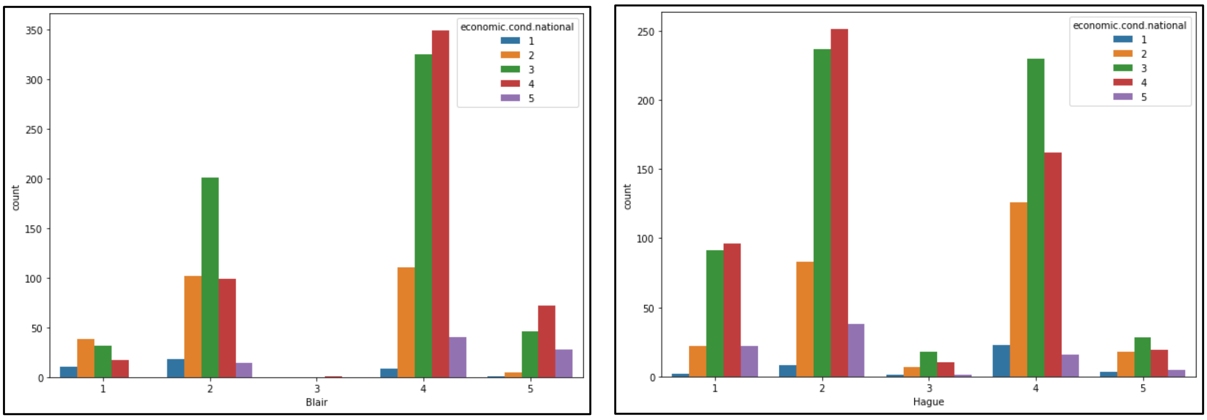
Categorical variables and numerical variables are also analysed using suitable visualization techniques.

Analysis of Blair and Hague with age variable

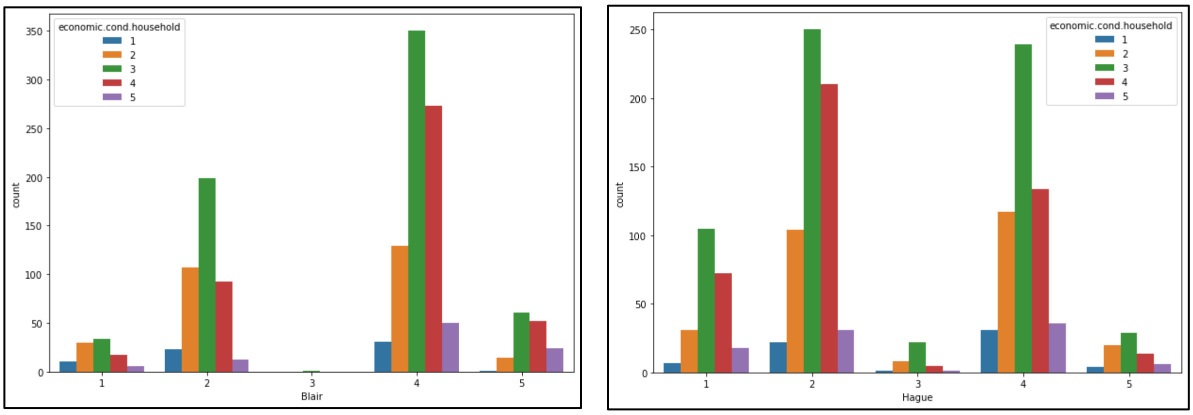


1.2.3. Boxplot, strip plot (Blair & Hague with age)

- The plot indicates that people above age 45 think that Blair is doing an excellent job. Blair have only a few votes in the (3) neutral assessment.
- People above age 50 think that Hague is doing an excellent job, above age 40 people thinks Hague is doing good job.

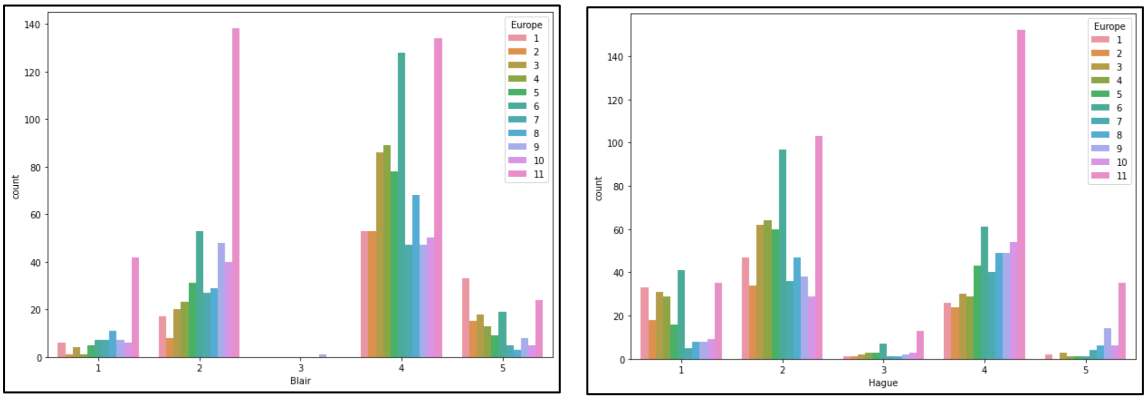


1.2.4. Count plot (Blair & Hague with economic.cond.national)



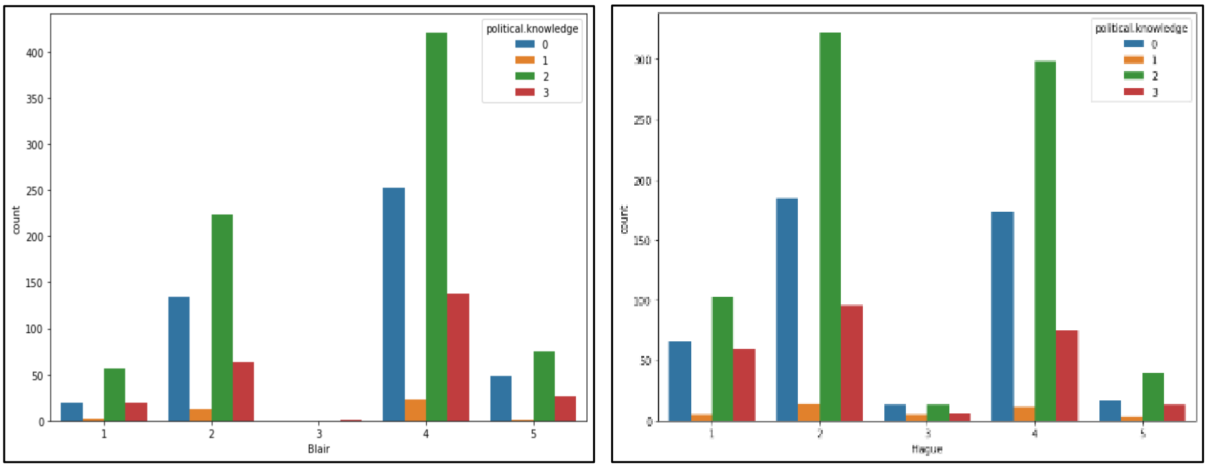
1.2.5. Count plot (Blair & Hague with economic.cond.household)

- From figure 1.2.4, Blair has really good points in the economic.cond.national than Hague.
- From figure 1.2.5, Blair has really good points in the economic.cond.household than Hague.



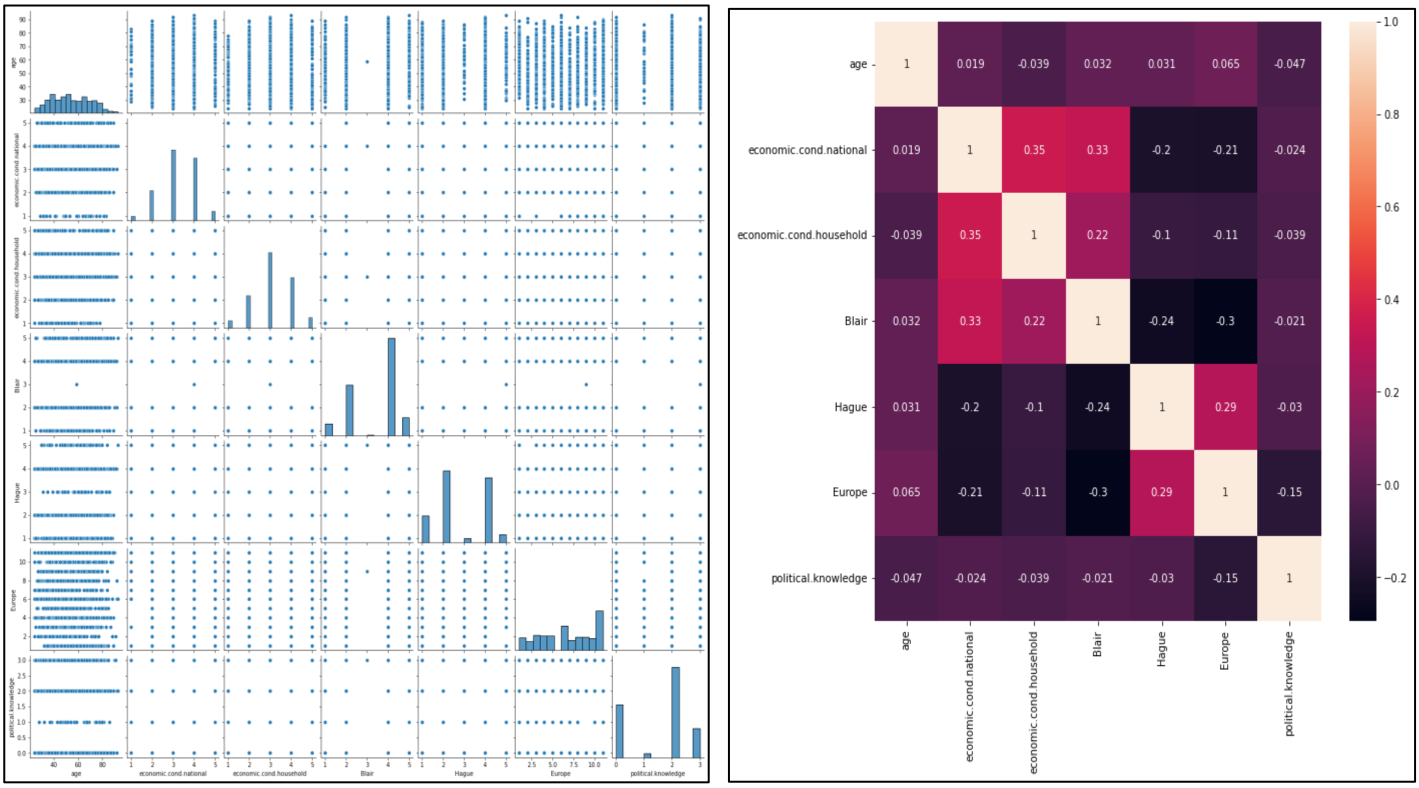
1.2.6. Count plot (Blair & Hague with 'Eurosceptic' sentiment)

- From figure 1.2.6, it is observed that most of the voters is having 'Eurosceptic' sentiment and supports Blair.



1.2.7. Count plot (Blair & Hague with Political knowledge)

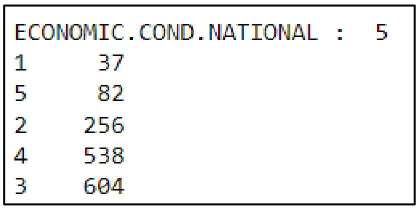
- From figure 1.2.7, it is observed that most of the voters have given points to Blair in terms of knowledge of parties' positions on European integration.
- Even though Hague got lower points than Blair, people have considered him to be knowledgeable in the parties' positions on European integration.



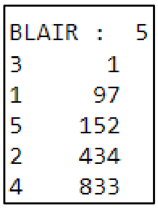
1.2.8. Pair plot & Heatmap (Continuous variables)

- From the pair plot and heatmap, it is observed that there is no multicollinearity between the independent variables. The independent variables are independent of each other.
- The highest positive correlation is found between the economic.cond.national and economic.cond.household at 35%.
- The highest negative correlation is found between the Europe and Blair at 30%.
- Hague with economic.cond.national and Blair have moderate negative correlation.
- Europe with economic.cond.national and Blair have moderate negative correlation.

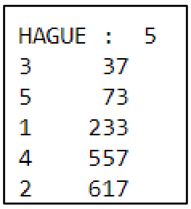
The number of data points of the features in the election dataset are as follows,



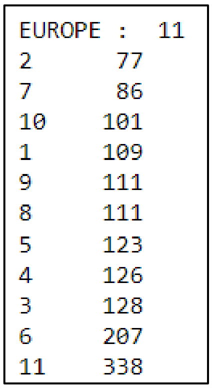
```
ECONOMIC.COND.NATIONAL :  5
1     37
5     82
2    256
4    538
3    604
```



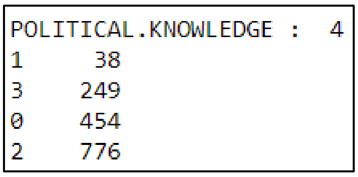
```
BLAIR :  5
3      1
1     97
5    152
2    434
4    833
```



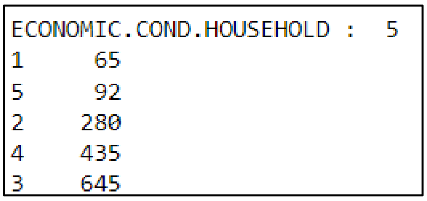
```
HAGUE :  5
3     37
5     73
1    233
4    557
2    617
```



```
EUROPE :  11
2      77
7      86
10    101
1     109
9     111
8     111
5     123
4     126
3     128
6     207
11    338
```



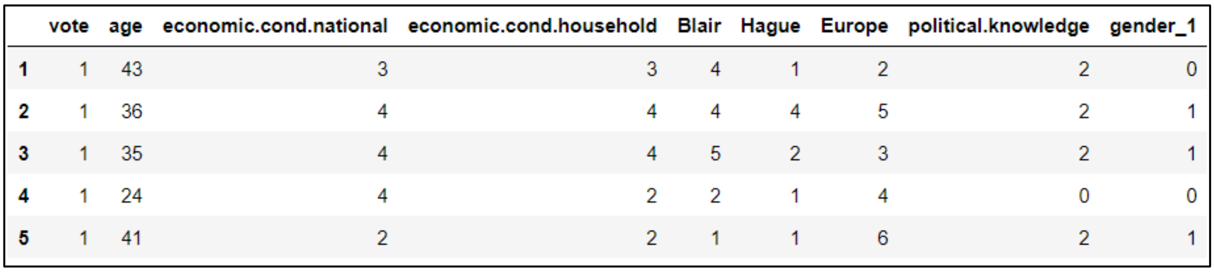
```
POLITICAL.KNOWLEDGE :  4
1     38
3    249
0    454
2    776
```



```
ECONOMIC.COND.HOUSEHOLD :  5
1     65
5     92
2    280
4    435
3    645
```

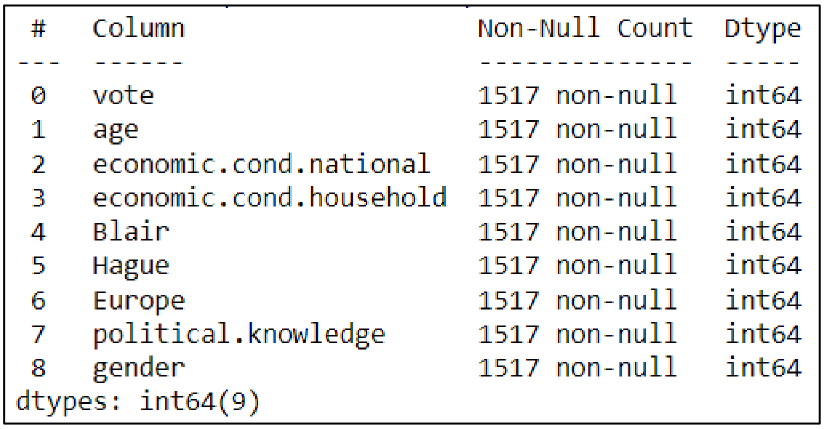
1.3) Encode the data (having string values) for Modelling. Is Scaling necessary here or not? Data Split: Split the data into train and test (70:30). The learner is expected to check and comment about the difference in scale of different features on the bases of appropriate measure for example std dev, 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. (pd.categorical().codes(),pd.get_dummies(drop_first=True))Data split, ratio defined for the split, train-test split should be discussed.

Encoded data frame:



	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_1
1	1	43	3	3	4	1	2	2	0
2	1	36	4	4	4	4	5	2	1
3	1	35	4	4	5	2	3	2	1
4	1	24	4	2	2	1	4	0	0
5	1	41	2	2	1	1	6	2	1

Encoded data info:



```
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   vote                     1517 non-null   int64
 1   age                      1517 non-null   int64
 2   economic.cond.national   1517 non-null   int64
 3   economic.cond.household  1517 non-null   int64
 4   Blair                    1517 non-null   int64
 5   Hague                    1517 non-null   int64
 6   Europe                   1517 non-null   int64
 7   political.knowledge      1517 non-null   int64
 8   gender                   1517 non-null   int64
dtypes: int64(9)
```

Train-test-split:

- Our model will use all the variables and 'vote' is the target variable. The train-test split is a technique for evaluating the performance of a machine learning algorithm. The procedure involves taking a dataset and dividing it into two subsets.

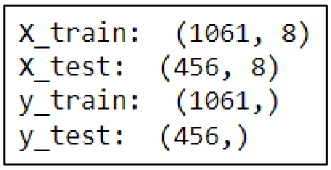
Train Dataset: It is used to train the model; it is used to fit the machine learning model.

Test Dataset: It is used to test the model; it is used to evaluate the fit machine learning model.

- The data is divided into 2 subsets, training and testing set. Earlier, we extracted the target variable ‘vote’ in a separate ‘y’ variable for subsets. Random state chosen as 1. Data split ratio is (70:30), 70% training data and 30% test data.

- 70% of the 'X-independent features' and 'y-target feature' from the data is used as train data to build the model and the remaining 30% of the 'X-independent features' and 'y-target feature' from the data will be used as test data to evaluate the built model.

Train-Test-Split Shape: (X = independent variables, y = target variable)



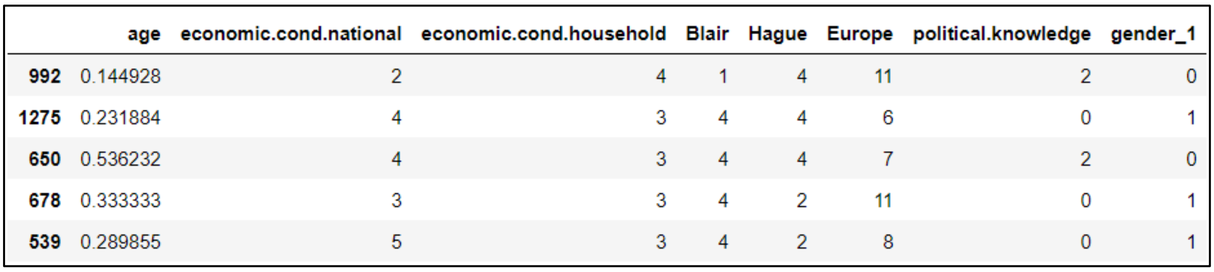
```
X_train:  (1061, 8)
X_test:  (456, 8)
y_train:  (1061,)
y_test:  (456,)
```

Scaling of dataset:

- The dataset contains features highly varying in magnitudes, units and range between the 'age' column and other columns. Since, most of the machine learning algorithms are using the ‘Euclidean distance’ between two data points in their computations, unscaled data becomes a problem.

- The ‘age’ features with high magnitudes will be weighed in a lot more in the distance-based calculations than the features with low magnitudes. To correct this, we need to bring ‘age’ features to the same level of magnitudes. This can be achieved by scaling using the MinMaxScaler technique to scale the data. Since other features are ordinal in nature, we decide to proceed without scaling them.

Train data after scaling:



	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_1
992	0.144928	2	4	1	4	11	2	0
1275	0.231884	4	3	4	4	6	0	1
650	0.536232	4	3	4	4	7	2	0
678	0.333333	3	3	4	2	11	0	1
539	0.289855	5	3	4	2	8	0	1

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

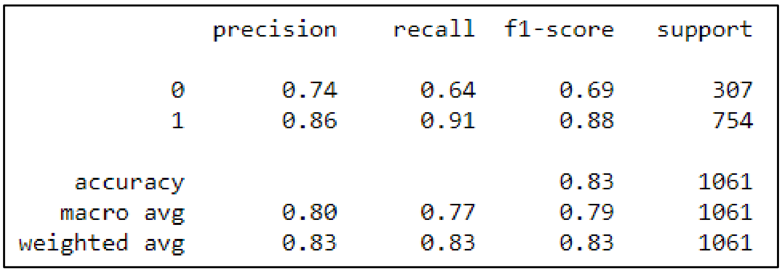
Logistic Regression:

The various important parameters of the logistic regression model are penalty, solver, max_iter, tol, etc.

Initially for building our model let us assign the following parameters: solver='newton-cg', max_iter=10000, penalty='none',verbose=True,n_jobs=2.We can optimize with best parameters by using GridSearchCV.

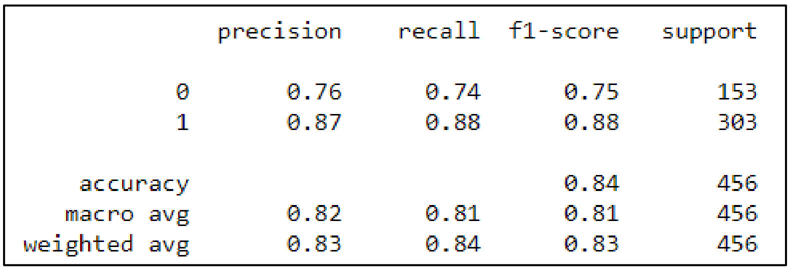
Accuracy - Training Data: 0.8312912346842601
Accuracy - Test Data **:** 0.8355263157894737

Classification report of Training data:



```
              precision    recall  f1-score   support

           0       0.74      0.64      0.69       307
           1       0.86      0.91      0.88       754

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

Classification report of Test data:



```
              precision    recall  f1-score   support

           0       0.76      0.74      0.75       153
           1       0.87      0.88      0.88       303

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

Observations:

Train data:

- **Accuracy: 83.12%**
- **Precision: 86%**
- **Recall: 91%**
- **FI-Score: 88%**

Test data:

- **Accuracy: 83.5%**
- **Precision: 87%**
- **Recall: 88%**
- **FI-Score: 88%**

Validness of model:

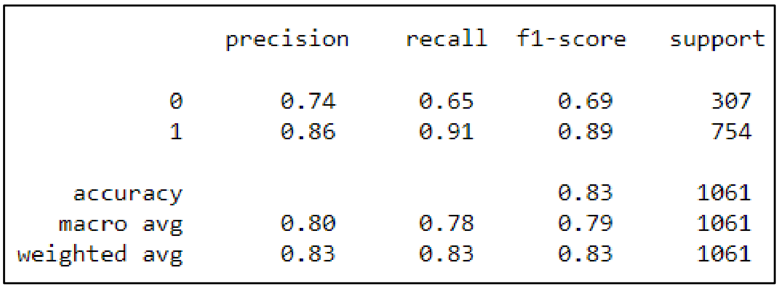
- The model is performing well on both the train and the test dataset, this concludes that the model is valid and is not over-fitted nor under-fitted.
- The accuracy of the test and train data is not very different
- For our model precision has improved and recall has reduced in the test data. But the change in both the parameters are very minimal(<5%).

Linear Discriminant Analysis Model:

- The various important parameters of the LDA model are solver, shrinkage. We will perform GridSearchCV later to find out the optimum parameter. For building the model now, we are assigning the parameters as default values.

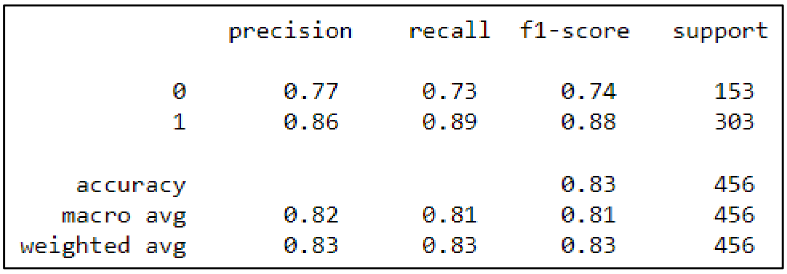
Accuracy - Train data: 0.8341187558906692
Accuracy - Test data: 0.8333333333333334

Classification report - Train data:



```
              precision    recall  f1-score   support

           0       0.74      0.65      0.69       307
           1       0.86      0.91      0.89       754

    accuracy                           0.83      1061
   macro avg       0.80      0.78      0.79      1061
weighted avg       0.83      0.83      0.83      1061
```

Classification report - Test data:



```
              precision    recall  f1-score   support

           0       0.77      0.73      0.74       153
           1       0.86      0.89      0.88       303

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

Observations:

Train data:

- **Accuracy: 83.41%**
- **Precision: 86%**
- **Recall: 91%**
- **FI-Score: 89%**

Test data:

- **Accuracy: 83.33%**
- **Precision: 86%**
- **Recall: 89%**
- **FI-Score: 88%**

Validness of model:

- The model is performing well on both the train and the test dataset, this concludes that the model is valid and is not over-fitted nor under-fitted.

- The accuracy of the test and train data is not very different
- For our model precision has remained same and recall has reduced in the test data. But the change in both the parameters are very minimal(<5%).

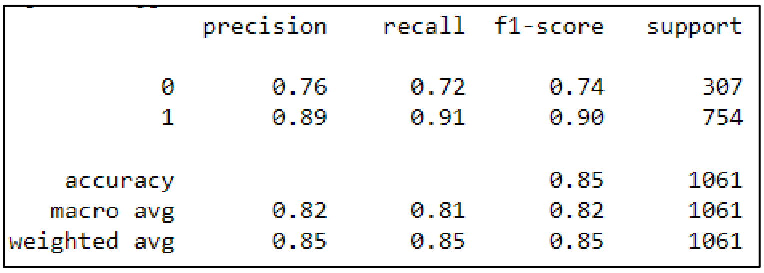
1.5) Apply KNN Model and Naïve Bayes Model. Interpret the inferences of each model. Successful implementation of each model. Logical reason behind the selection of different values for the parameters involved in each model. Calculate Train and Test Accuracies for each model. Comment on the validness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

K-Nearest Neighbour Model:

The K value, weights, algorithm is important in the KNN model and by default it will be k=5. Let's build a model by using default value and optimize the model after observing the performance.

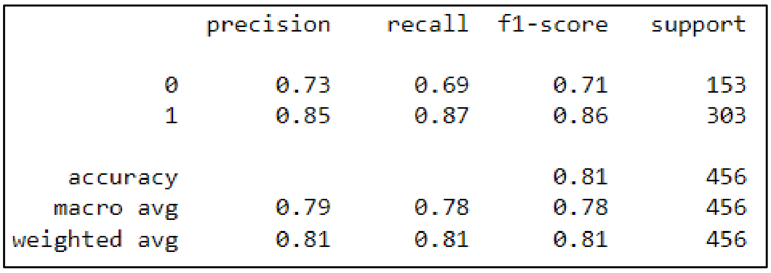
Accuracy - Train data: 0.8510838831291234
Accuracy - Test data: 0.8114035087719298

Classification report - Train data:



```
              precision    recall  f1-score   support

           0       0.76      0.72      0.74       307
           1       0.89      0.91      0.90       754

    accuracy                           0.85      1061
   macro avg       0.82      0.81      0.82      1061
weighted avg       0.85      0.85      0.85      1061
```

Classification report - Test data:



```
              precision    recall  f1-score   support

           0       0.73      0.69      0.71       153
           1       0.85      0.87      0.86       303

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

Observations:

Train data:

- **Accuracy: 85.11%**
- **Precision: 89%**
- **Recall: 91%**
- **FI-Score: 90%**

Test data:

- **Accuracy: 81.14%**
- **Precision: 85%**
- **Recall: 87%**
- **FI-Score: 86%**

Validness of model:

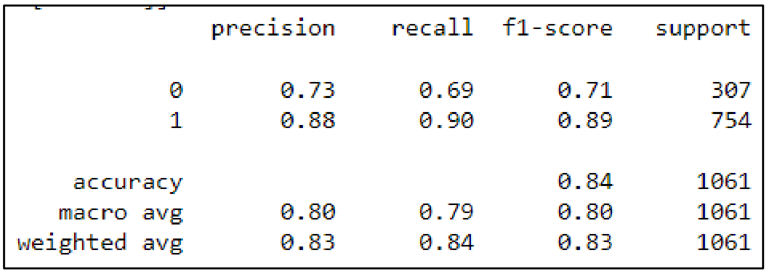
- The model performed well with the data.
- In this model there is a slightly good performance on the training data than the test data. This may indicate a slightly overfitted model.
- Even though the model performs better on the train data, this slight over-fit issue can be solved by tuning the model with optimal parameters.

Naive Bayes Model:

The NB model is based on applying the bayes theorem with an assumption that the predictor variables are independent from each other. This assumption may not hold true in real-life scenarios.

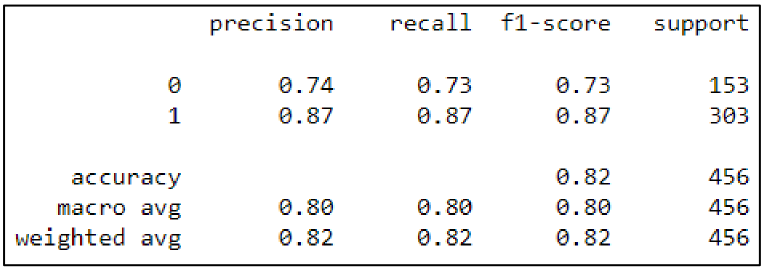
Accuracy - Train data: 0.8350612629594723
Accuracy - Test data: 0.8223684210526315

Classification report - Train data:



```
              precision    recall  f1-score   support

           0       0.73      0.69      0.71       307
           1       0.88      0.90      0.89       754

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

Classification report - Test data:



```
              precision    recall  f1-score   support

           0       0.74      0.73      0.73       153
           1       0.87      0.87      0.87       303

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

Observations:

Train data:

- **Accuracy: 84%**
- **Precision: 88%**
- **Recall: 90%**
- **FI-Score: 89%**

Test data:

- **Accuracy: 82%**
- **Precision: 87%**
- **Recall: 87%**
- **FI-Score: 87%**

Validness of model:

- The model is performing well on both the train and the test dataset, this concludes that the model is valid and is not over-fitted nor under-fitted.
- The accuracy of the test and train data is only at 2% difference.
- For our model precision has no significant change and recall has reduced 3% in the test data. But the change in both the parameters are very minimal.

1.6) Model Tuning, Bagging and Boosting. 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.

Model Tuning

Tuning is the process of enhancing the performance of the model without the over-fitting. In ML algorithms this can be achieved by selecting suitable 'hyper-parameters'

Grid search is one of the common methods to optimize the parameters. Here a set of parameters are defined and then the performance of each combination is evaluated, using cross validation. Models such as Bagging, Boosting, Gradient boosting, are prone to over fitting of data.

Bagging

Bagging is an ensemble technique. This model combines several base models to get the optimal model. Bagging is designed to improve the performance of the ML algorithms used in classification or regression. Bagging is a parallel method.

Bagging trains the base classifier with the training set parallelly by drawing them randomly. In our bagging model we are using random forest as the base model. Hyper parameters of the random forest models are

- max_depth = 10
- max_features = 10
- min_samples_leaf = 15
- min_samples_split= 30
- n_estimators = 100

Accuracy - Train data: 84.449
Accuracy - Test data: 82.237

Observation:

- The model is performing well with good accuracy on both the train and the test dataset, this concludes that the model is valid and is not over-fitted nor under-fitted.

Boosting

Boosting is also an ensemble technique. As the name suggests this technique makes the weak learners to strong learners. This is a sequential method where the learner is drawn, and it is used as input for the next learner and goes on like this. The prediction which was made wrong is given a higher weight and a correct prediction is made. But here the correctly prediction will lose weight. Here we apply Ada boosting and Gradient boosting on our dataset.

ADA Boosting

For the ADA boosting model, the hyper-parameters involved in model optimization are below. After performing the GridsearchCV, the best parameters are

- Algorithm = SAMME
- n_estimators =1000

Accuracy - Train data: 83.7%
Accuracy - Test data: 80.9%

Observation:

- The model is performing well with good accuracy on both the train and the test dataset, this concludes that the model is valid and is not over-fitted nor under-fitted.

Gradient Boosting

The gradient boosting technique is similar to ADA boosting. It works by adding the misidentified predictors and under-fitted predictions to the ensemble, this ensures the errors identified previously are corrected. The hyper parameters of the gradient boosting are criterion, loss, n_estimators, max_features, min_samples_split.

Accuracy - Train data: 89.25%
Accuracy - Test data: 84.33%

Observation:

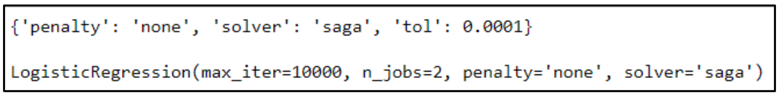
- The model is performing well with good accuracy on both the train and the test dataset, this concludes that the model is valid.

Below are the best 'hyper parameters' for different models used in our dataset,

Logistic Regression Model Tuning:

The logistic model is tuned, and the best hyper parameters and output is below,

Best parameters:

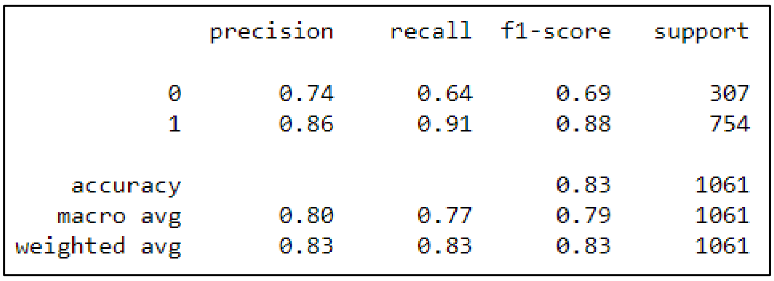


```
{'penalty': 'none', 'solver': 'saga', 'tol': 0.0001}

LogisticRegression(max_iter=10000, n_jobs=2, penalty='none', solver='saga')
```

Accuracy - Train data: 83.13

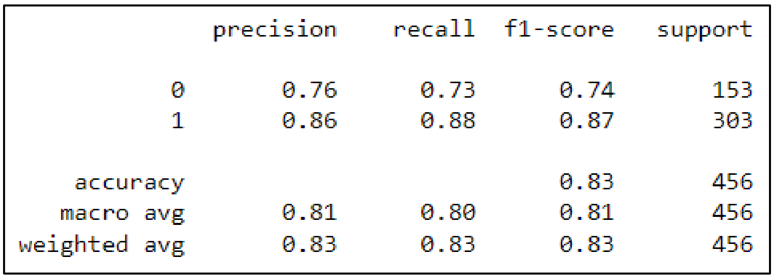
Accuracy - Test data: 83.11

Classification report - Train data:



```
              precision    recall  f1-score   support

           0       0.74      0.64      0.69       307
           1       0.86      0.91      0.88       754

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

Classification report - Test data:



```
              precision    recall  f1-score   support

           0       0.76      0.73      0.74       153
           1       0.86      0.88      0.87       303

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

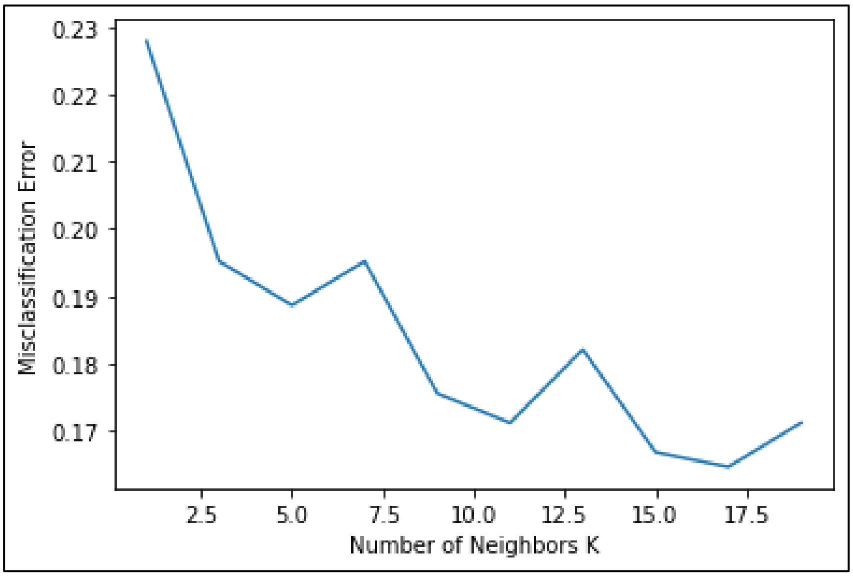
Observation:

- By referring the output of the base model and comparing it with the output of the tuned logistic model, it is found that no significant improvement is achieved.
- The overall output of the model is high and there is no over-fitting or under-fitting. Therefore, both the base and tuned models are equally good models.

K-Nearest Neighbour Model Tuning:

The KNN model is tuned, and the best hyper parameters and output is below,

Best parameters:



From the plot, the number of neighbours is found to be 17 as it gives the least MCE value.

n_neighbors=17

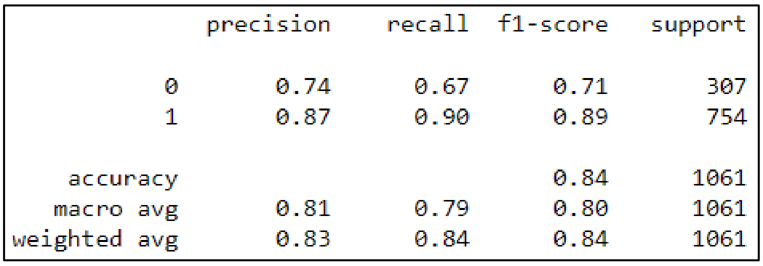
weights='uniform',

algorithm='auto',

leaf_size=30,

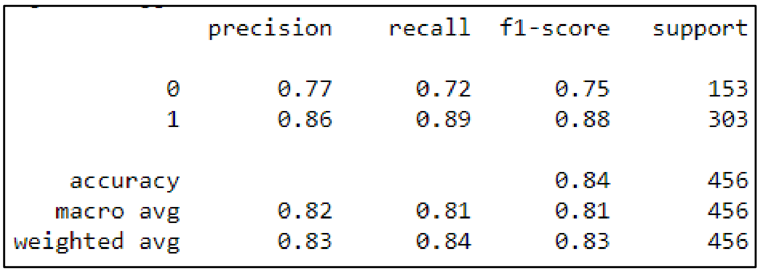
Accuracy - Train data: 83.8%
Accuracy - Test data: 83.55%

Classification report - Train data:



```
              precision    recall  f1-score   support

           0       0.74      0.67      0.71       307
           1       0.87      0.90      0.89       754

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

Classification report - Test data:



```
              precision    recall  f1-score   support

           0       0.77      0.72      0.75       153
           1       0.86      0.89      0.88       303

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

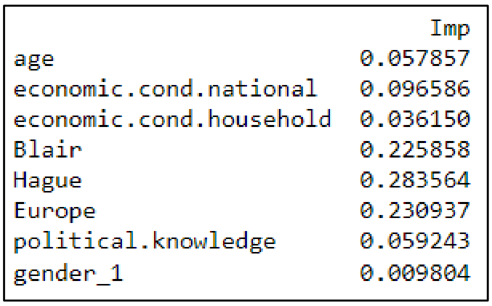
Observations:

- There is no over-fitting or under-fitting in the tuned KNN model. Overall, it is a good model.
- The regular KNN model was over-fitted. But model tuning has helped the model to recover from over-fitting.
- The values are better in the tuned KNN model. Therefore, the tuned KNN model is a better model.

Random Forest Model Tuning:

The Random Forest model is tuned, and the best hyper parameters are used to get the below output,

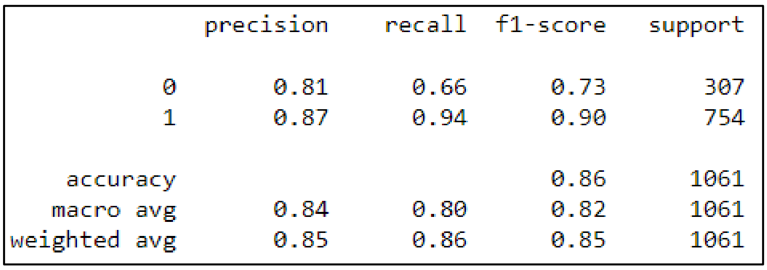
Feature importance's:



```
                              Imp
age                      0.057857
economic.cond.national   0.096586
economic.cond.household  0.036150
Blair                    0.225858
Hague                    0.283564
Europe                   0.230937
political.knowledge      0.059243
gender_1                 0.009804
```

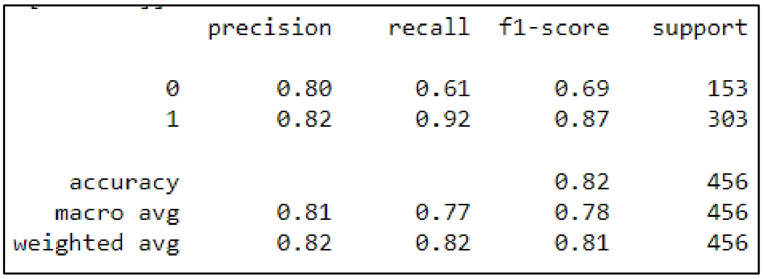
Accuracy - Train data: 85.8%
Accuracy - Test data: 81.8%

Classification report - Train data:



```
              precision    recall  f1-score   support

           0       0.81      0.66      0.73       307
           1       0.87      0.94      0.90       754

    accuracy                           0.86      1061
   macro avg       0.84      0.80      0.82      1061
weighted avg       0.85      0.86      0.85      1061
```

Classification report - Test data:



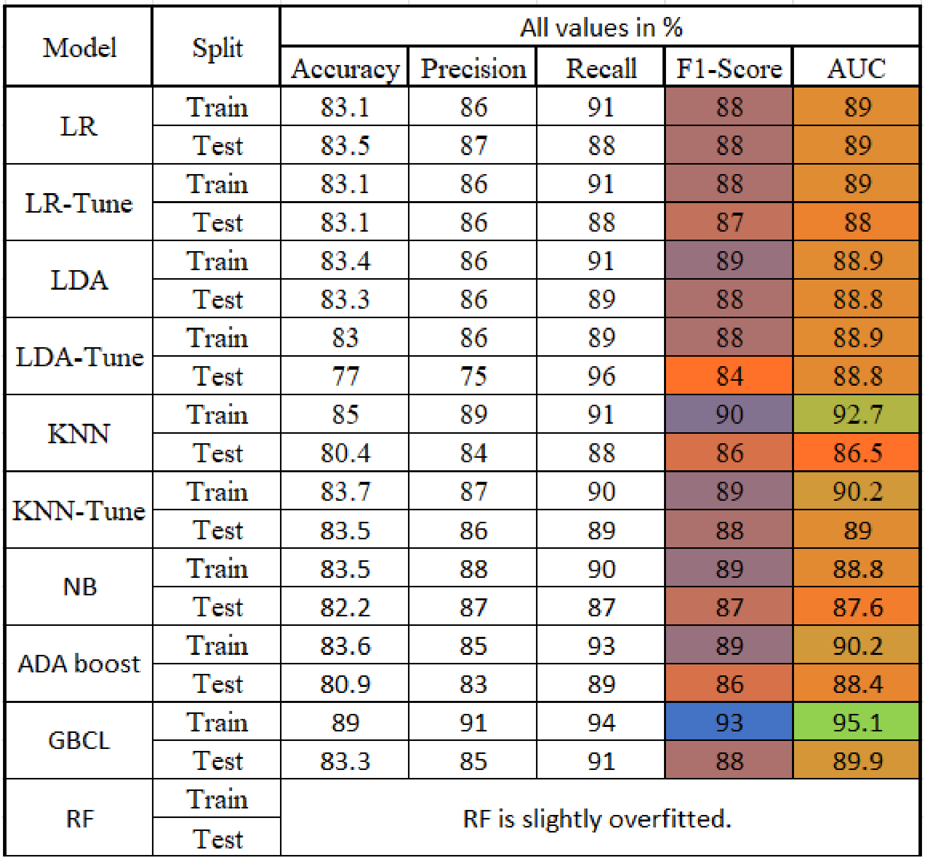
```
              precision    recall  f1-score   support

           0       0.80      0.61      0.69       153
           1       0.82      0.92      0.87       303

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

Observations:

- There is not over-fitted or under-fitting in the tuned RF model. Overall, it is a good model.
- The RF model is having about 4% difference in the accuracy of the training and test data. Since this difference is less than 5%, we are concluding this model to be valid in predicting the target variable.

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 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.

Performance metrics in a structured table,

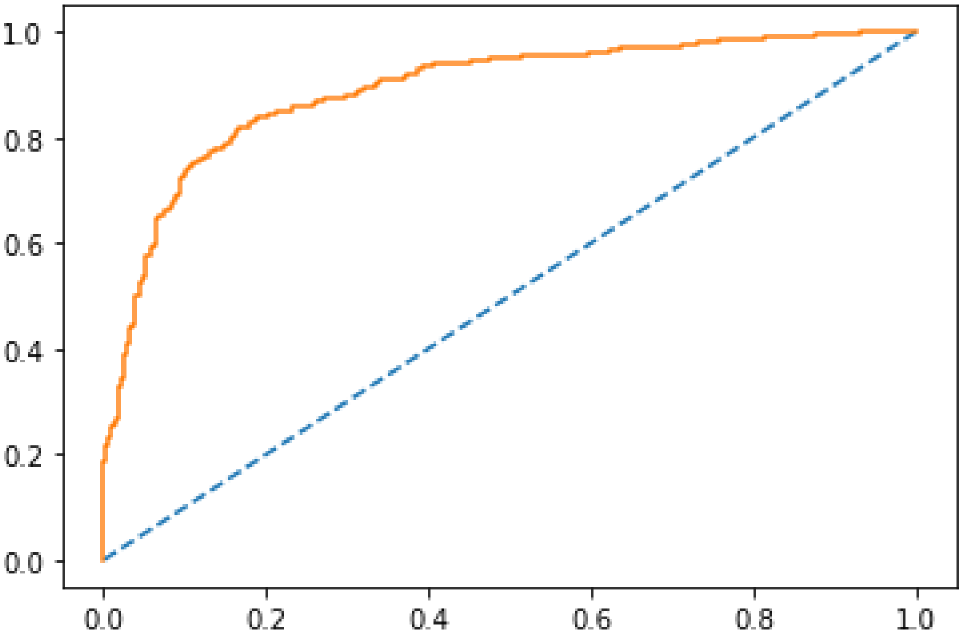


Model	Split	All values in %				
		Accuracy	Precision	Recall	F1-Score	AUC
LR	Train	83.1	86	91	88	89
	Test	83.5	87	88	88	89
LR-Tune	Train	83.1	86	91	88	89
	Test	83.1	86	88	87	88
LDA	Train	83.4	86	91	89	88.9
	Test	83.3	86	89	88	88.8
LDA-Tune	Train	83	86	89	88	88.9
	Test	77	75	96	84	88.8
KNN	Train	85	89	91	90	92.7
	Test	80.4	84	88	86	86.5
KNN-Tune	Train	83.7	87	90	89	90.2
	Test	83.5	86	89	88	89
NB	Train	83.5	88	90	89	88.8
	Test	82.2	87	87	87	87.6
ADA boost	Train	83.6	85	93	89	90.2
	Test	80.9	83	89	86	88.4
GBCL	Train	89	91	94	93	95.1
	Test	83.3	85	91	88	89.9
RF	Train	RF is slightly overfitted.				
	Test					

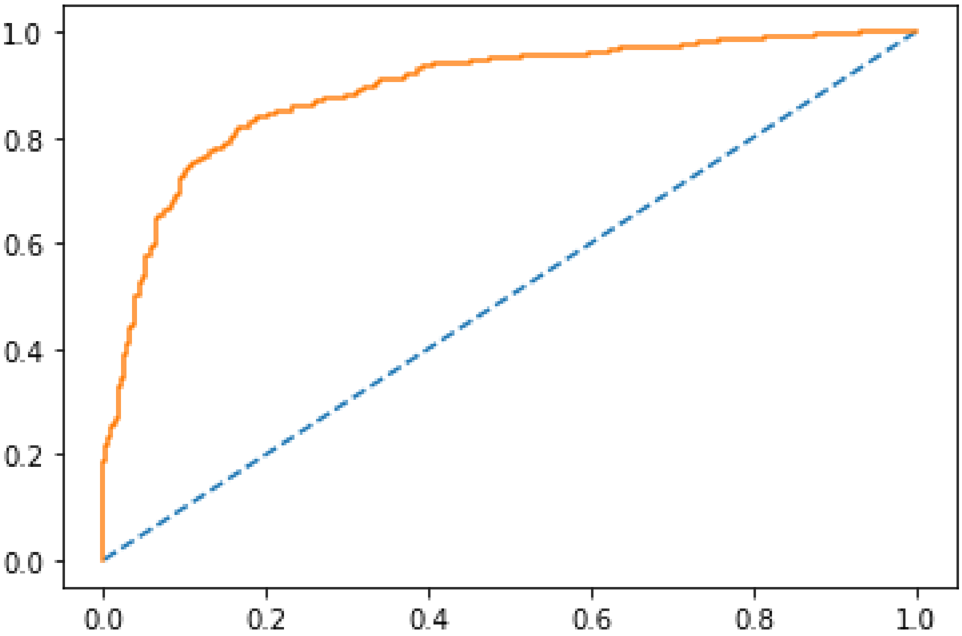
From the table it is clear that the tuned model is working well than the base model. Therefore, lets visualize the confusion matrix and AUC-ROC curve for these models to confirm the same.

Logistic Regression model

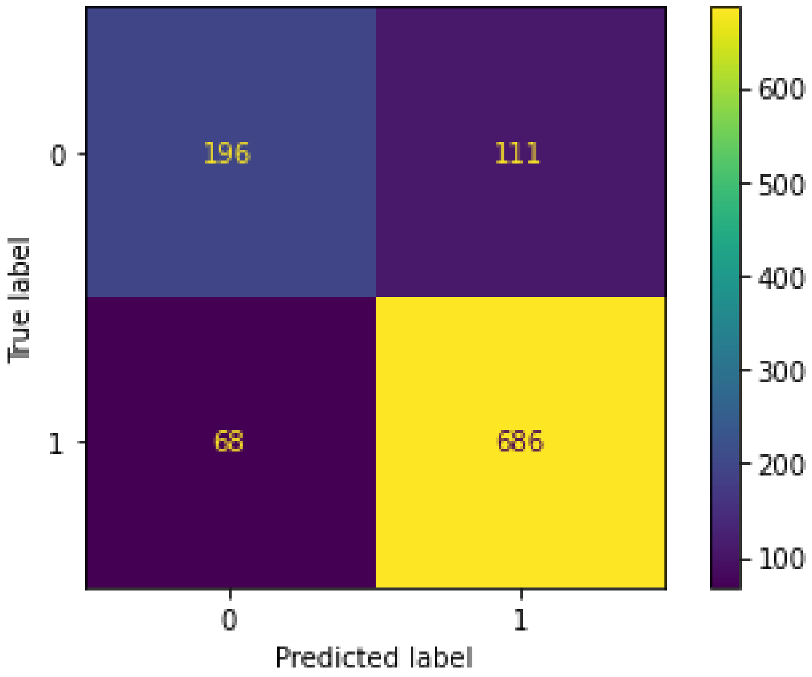
AUC and ROC for the training data¶



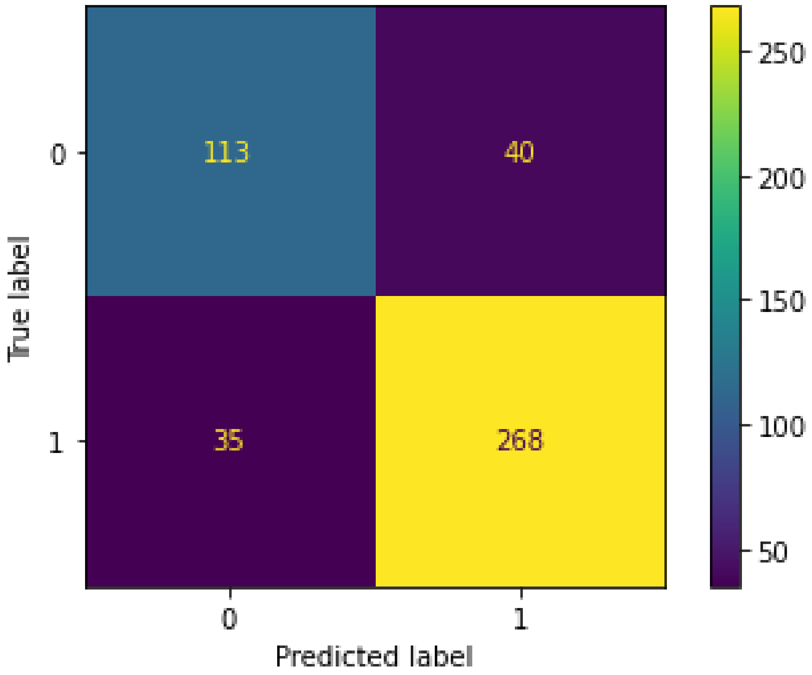
AUC and ROC for the test data



Confusion Matrix for the training data

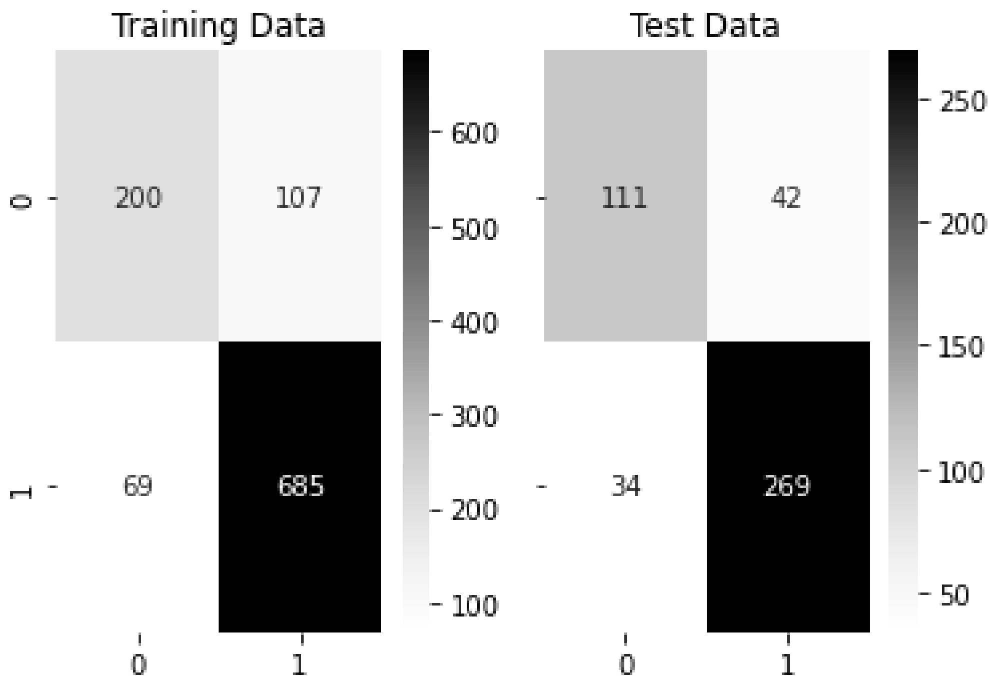


Confusion Matrix for test data

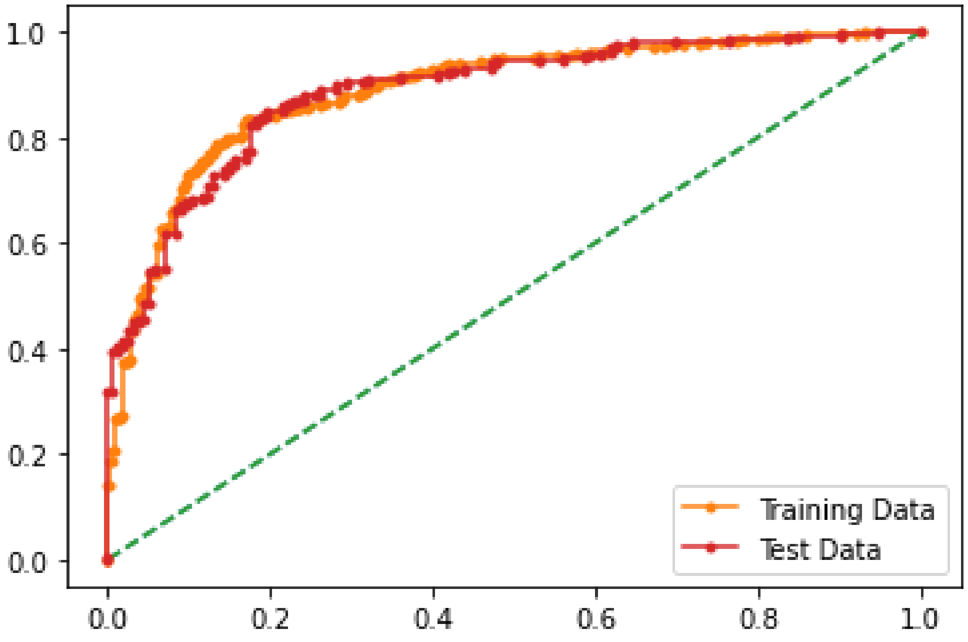


Linear Discriminant Analysis

Training Data and Test Data Confusion Matrix Comparison

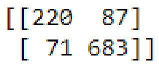


AUC Curve for the Training and Testing Data

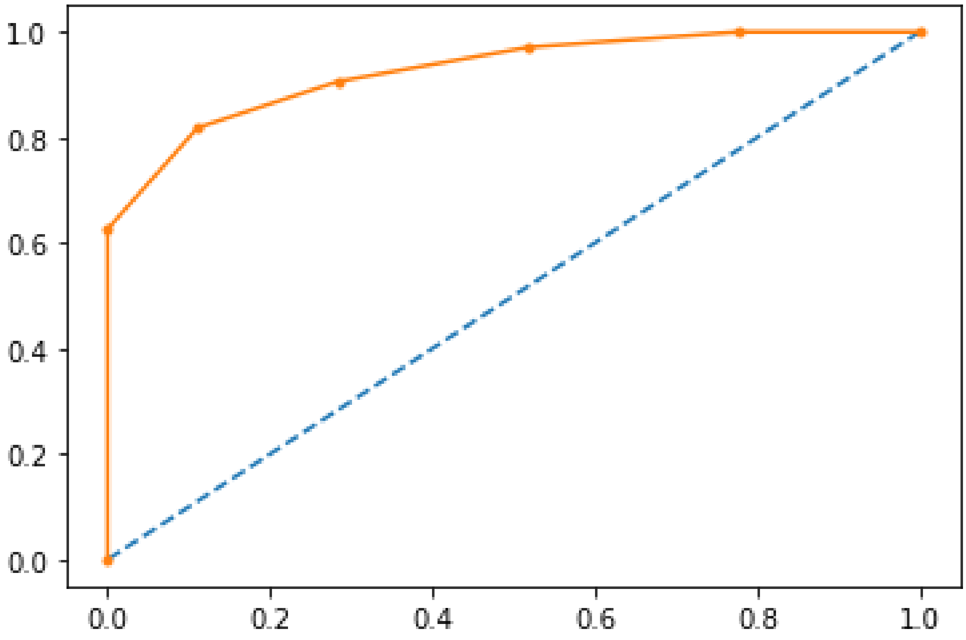


KNN model

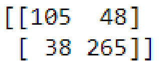
AUC and ROC for the training data , Performance Matrix on training data



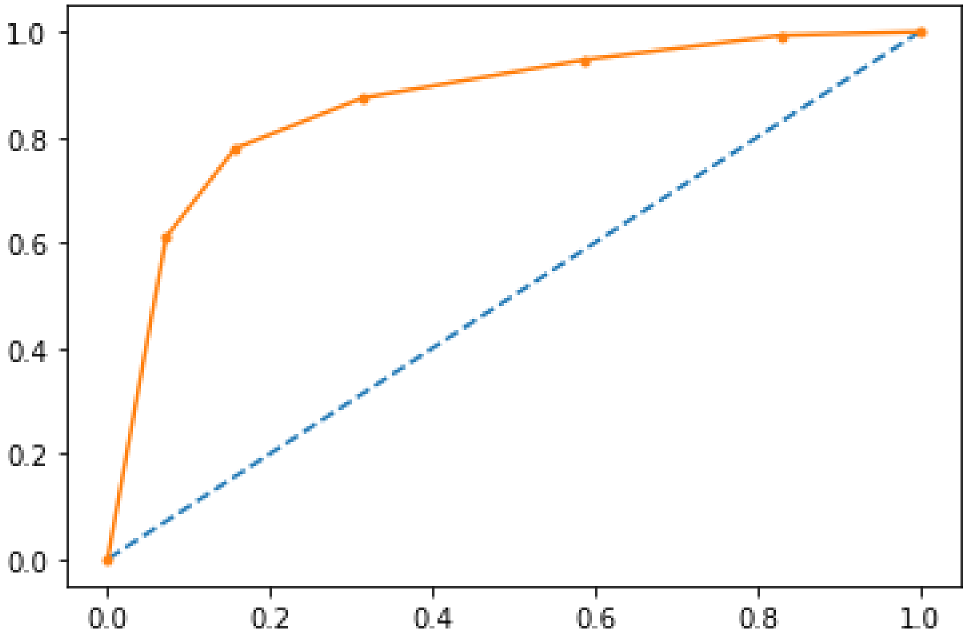
```
[[220  87]
 [ 71 683]]
```



Performance Matrix on test data, AUC and ROC for the test data

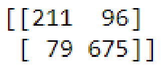


```
[[105  48]
 [ 38 265]]
```



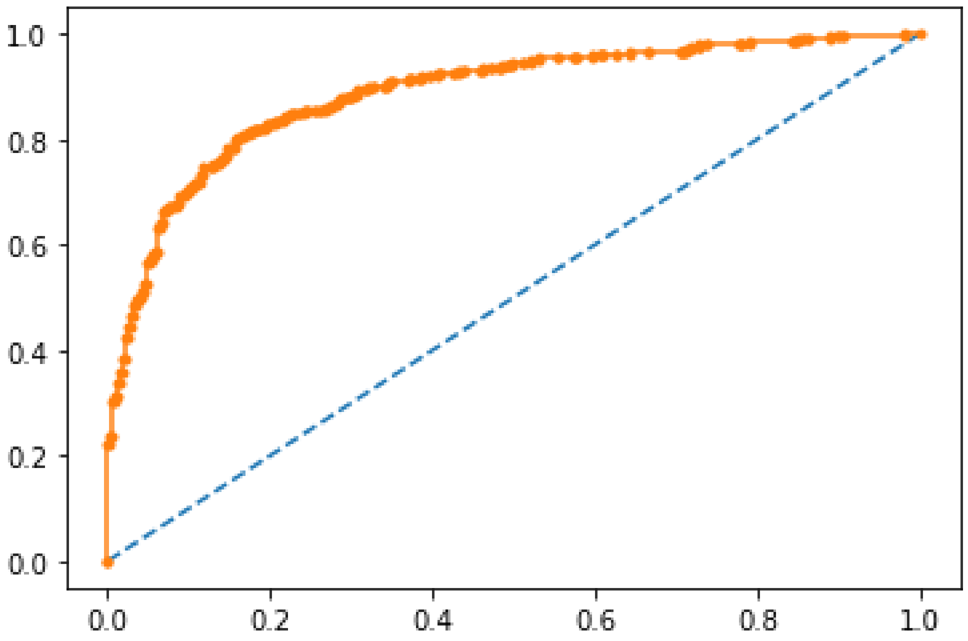
Gaussian Naive Bayes

Confusion matrix train data:

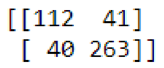


```
[[211  96]
 [ 79 675]]
```

AUC and ROC for the training data:

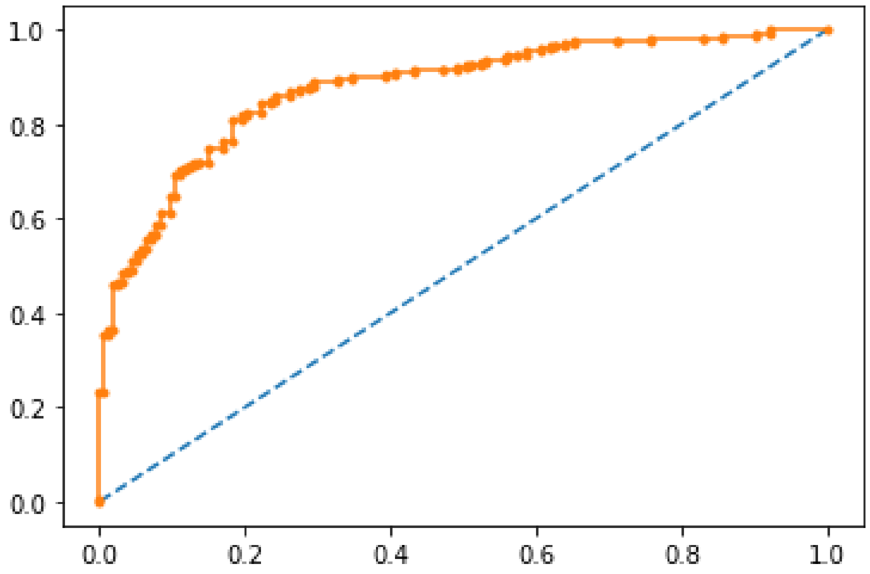


Confusion matrix test data:



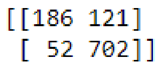
```
[[112  41]
 [ 40 263]]
```

AUC and ROC for the test data:



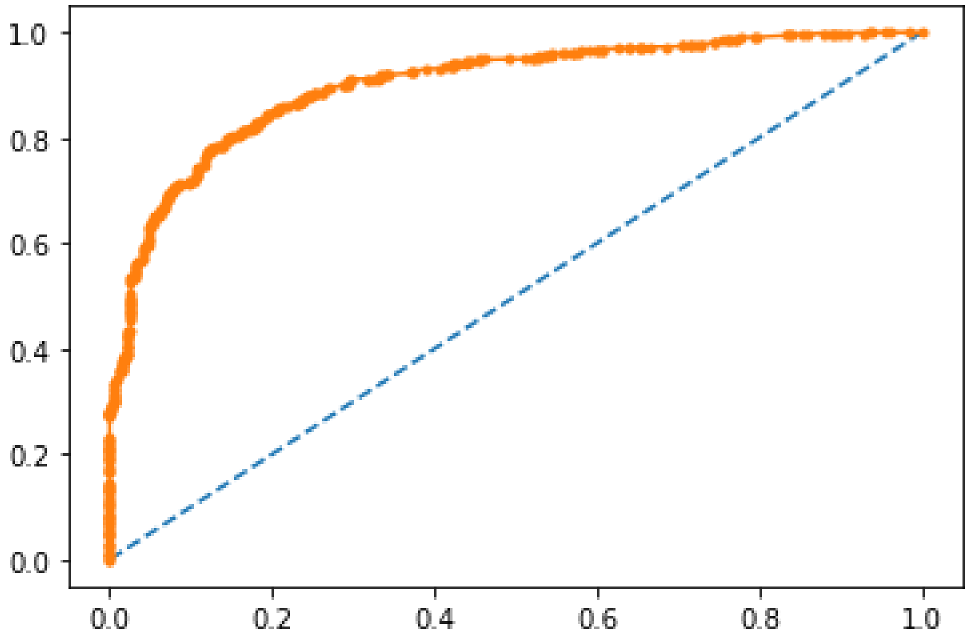
Ada boosting

Confusion matrix training data:

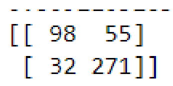


```
[[186 121]
 [ 52 702]]
```

AUC and ROC for the training data:

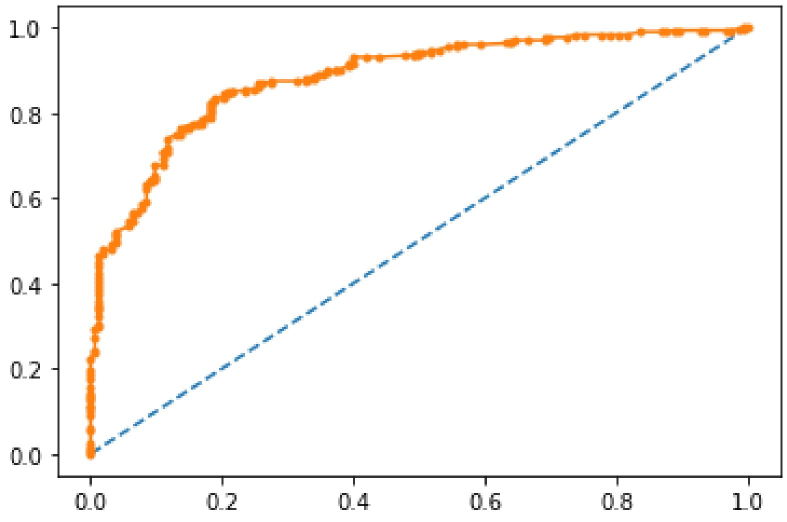


Confusion matrix test data:



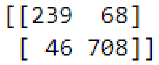
```
[[ 98  55]
 [ 32 271]]
```

AUC and ROC for the test data:



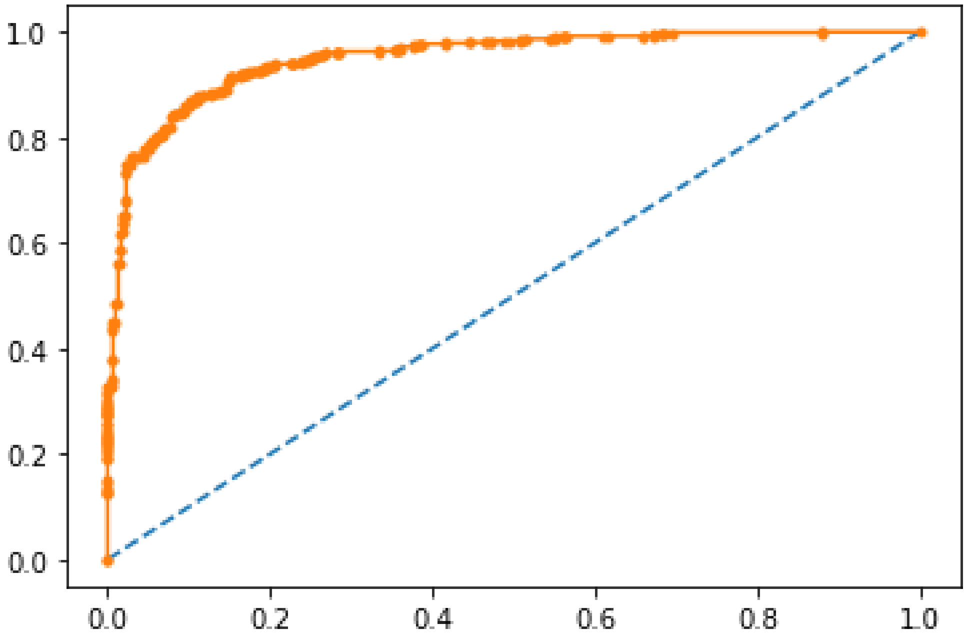
Gradient Boosting

Confusion matrix training data:

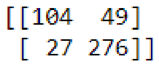


```
[[239  68]
 [ 46 708]]
```

AUC and ROC for the training data:

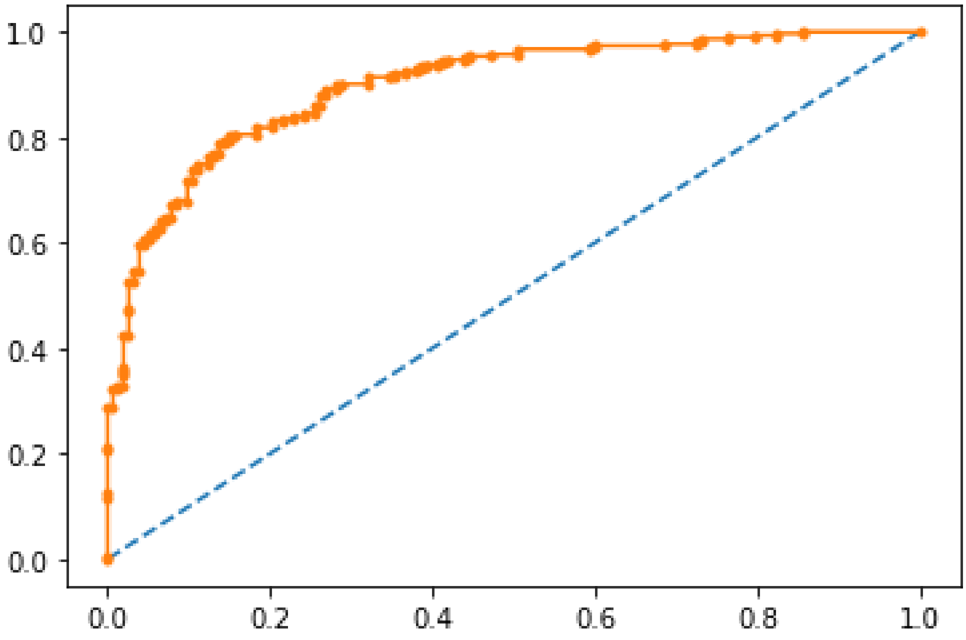


Confusion matrix test data:



```
[[104  49]
 [ 27 276]]
```

AUC and ROC for the test data:

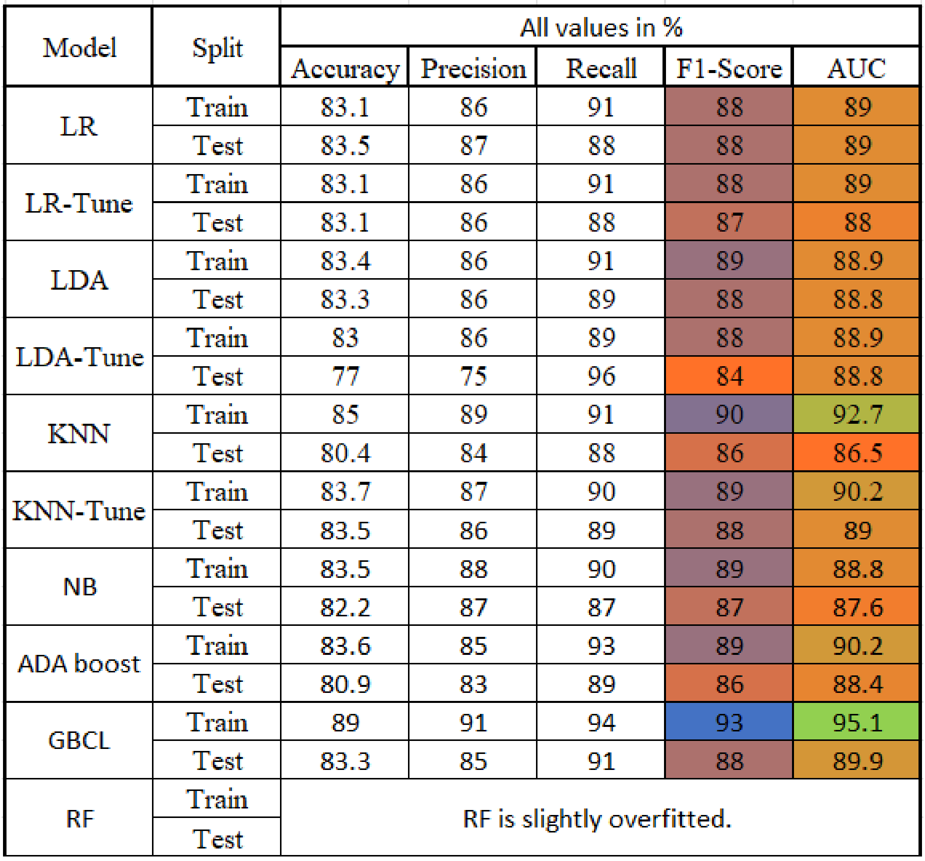


Conclusion:

- There is no under-fitting or over-fitting in any of the tuned models.
- All the tuned models have high values, and every model is good. But as we can see, the most consistent tuned model in both train and test data is the Gradient Boost model.

- The tuned gradient boost model performs the best with 93% accuracy score in train and 88% accuracy score in test. Also, it has the best AUC score of 95.1% in train and 89.9% in test data which is the highest of all the models.

- It also has a precision score of 91% and recall of 94%. So, we conclude that Gradient Boost Tuned model is the best/optimized model.



Model	Split	All values in %				
		Accuracy	Precision	Recall	F1-Score	AUC
LR	Train	83.1	86	91	88	89
	Test	83.5	87	88	88	89
LR-Tune	Train	83.1	86	91	88	89
	Test	83.1	86	88	87	88
LDA	Train	83.4	86	91	89	88.9
	Test	83.3	86	89	88	88.8
LDA-Tune	Train	83	86	89	88	88.9
	Test	77	75	96	84	88.8
KNN	Train	85	89	91	90	92.7
	Test	80.4	84	88	86	86.5
KNN-Tune	Train	83.7	87	90	89	90.2
	Test	83.5	86	89	88	89
NB	Train	83.5	88	90	89	88.8
	Test	82.2	87	87	87	87.6
ADA boost	Train	83.6	85	93	89	90.2
	Test	80.9	83	89	86	88.4
GBCL	Train	89	91	94	93	95.1
	Test	83.3	85	91	88	89.9
RF	Train	RF is slightly overfitted.				
	Test					

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.

Insights:

1. Labour party has more than double the votes of conservative party.
2. Most number of people have given a score of 3 and 4 for the national economic condition.
3. Most number of people have given a score of 3 and 4 for the household economic condition.
4. Blair has higher number of votes than Hague and the scores are much better for Blair than for Hague.
5. On a scale of 0 to 3, about 30% of the total population has zero knowledge about politics/parties.
6. People who have higher Eurosceptic sentiment, has voted for the conservative party and lower the Eurosceptic sentiment, higher the votes for Labour party.
7. All models performed well on training data set as well as test data set. The tuned models have performed better than the regular models.
8. There is no over-fitting in any model except Random Forest. Gradient Boosting model tuned is the best/optimized model.

Recommendations:

9. Gathering more data will also help in training the models and thus improving the predictive powers.
10. We can create a function for all the models to predict the outcome in sequence.
11. This will help in understanding the probability of what the outcome will be.
12. Parametric tuning is important as it helped in optimizing the models and achieves good predictions.

Problem 2:

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

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

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

1. **1941-Roosevelt**

   The number of characters in this speech: 7571
   The number of words in this speech   : 1536
   The number of sentences in this speech: 68

2. **1961-Kennedy**

   The number of characters in this speech: 7618
   The number of words in this speech   : 1546
   The number of sentences in this speech: 52

3. **1973-Nixon**

   The number of characters in this speech: 9991
   The number of words in this speech   :   2028
   The number of sentences in this speech: 69

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

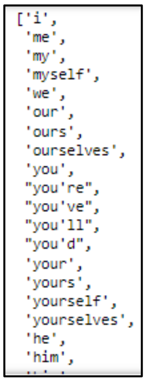
The stop words are words which do not have any importance in understanding the sentiment of the text data and not useful in terms of building the ML algorithm. These stop words can be removed from our dataset by using the stopwords package available in the natural language tool kit. This stopword library consists of some of the commonly used stop words like ‘i’, ‘me’, ‘myself’, ‘we’, ‘u’, etc. The language for which the stopwords are required needs to be specified, here we are using English language.

Before removing stop words (word count):

Word count before removing stop words in Roosevelt speech is 1536 words.
Word count before removing stop words in Kennedy speech is 1546 words.
Word count before removing stop words in Nixon speech is 2028 words.

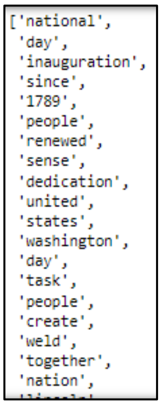
After removing stop words (word count):

Word count after removing stop words in Roosevelt speech is 632 words.
Word count after removing stop words in Kennedy speech is 697 words.
Word count after removing stop words in Nixon speech is 836 words.

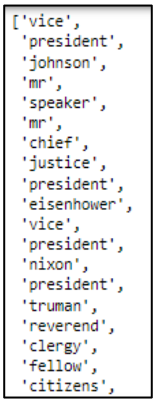


```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
```

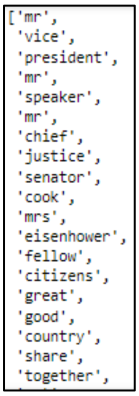
2.2.1. Stop words



```
['national',
 'day',
 'inauguration',
 'since',
 '1789',
 'people',
 'renewed',
 'sense',
 'dedication',
 'united',
 'states',
 'washington',
 'day',
 'task',
 'people',
 'create',
 'weld',
 'together',
 'nation',
```



```
['vice',
 'president',
 'johnson',
 'mr',
 'speaker',
 'mr',
 'chief',
 'justice',
 'president',
 'eisenhower',
 'vice',
 'president',
 'nixon',
 'president',
 'truman',
 'reverend',
 'clergy',
 'fellow',
 'citizens',
```



```
['mr',
 'vice',
 'president',
 'mr',
 'speaker',
 'mr',
 'chief',
 'justice',
 'senator',
 'cook',
 'mrs',
 'eisenhower',
 'fellow',
 'citizens',
 'great',
 'good',
 'country',
 'share',
 'together',
```

2.2.2. Stop words removed from speeches (Roosevelt, Kennedy, Nixon)

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

Each president might have used a word repetitively either subconsciously or consciously to make their speech effective and reach the people. Let us find the words which are used for the greatest number of times by the presidents,

The top three words occurring the greatest number of times in **1941-Roosevelt Speech** are

1. nation-12 (most occurring),
2. know-10 (2nd most occurring),
3. spirit-9, democracy-9, life-9 (three words are in 3rd most occurring)

The top three words occurring the greatest number of times in **1961-Kennedy Speech** are

1. let - 16 (most occurring),
2. us -12 (2nd most occurring),
3. sides-8 world-8 (two words are in 3rd most occurring)

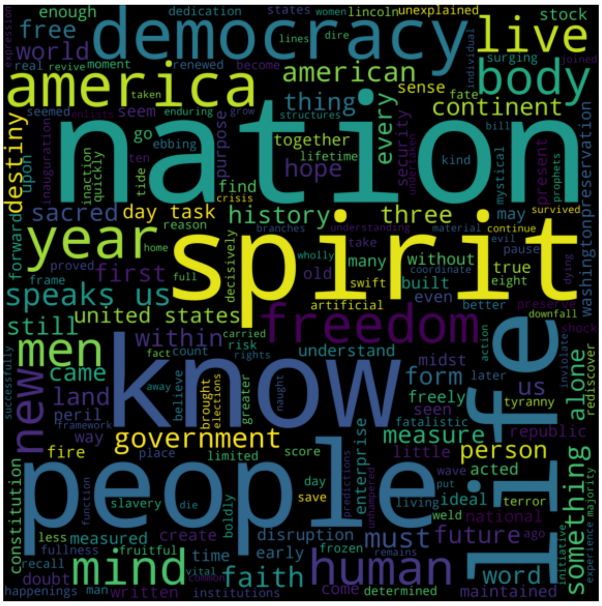
The top three words occurring the greatest number of times in **1973-Nixon Speech are**

1. us-26 (most occurring),
2. let -22 (2nd most occurring),
3. america-21 (3rd most occurring)

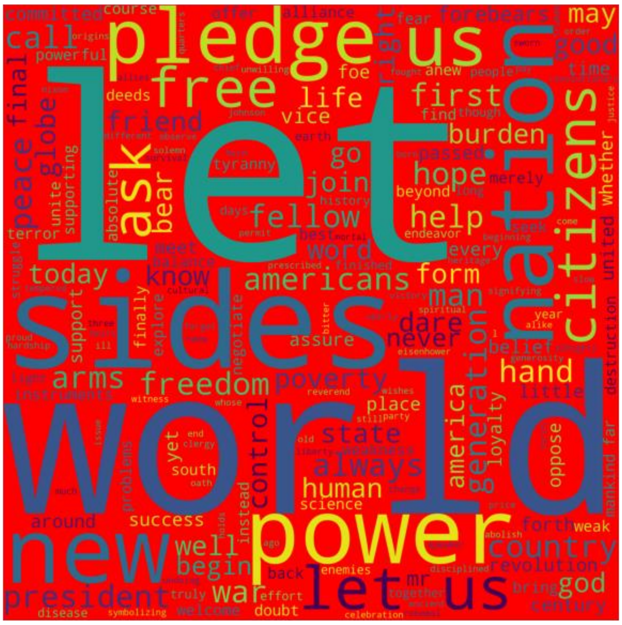
We observe that there are some common (let, us) words between President **Kennedy's** and **Nixon's speeches.**

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

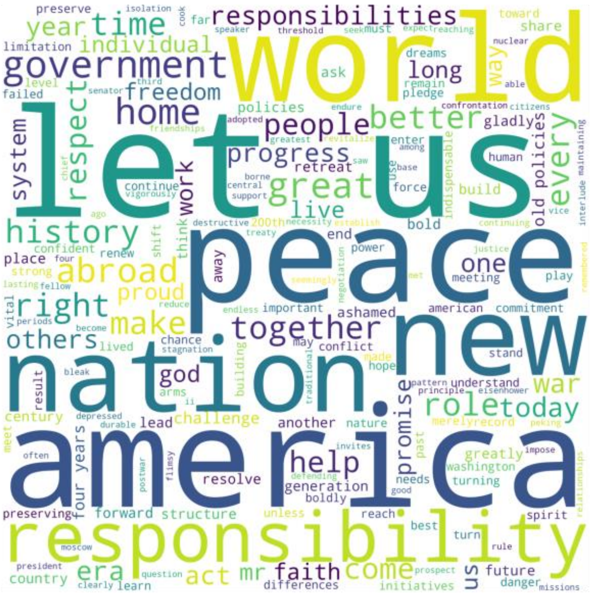
In text analysis, word cloud is a visualization technique used for plotting the words based on its importance and its frequency. From this we get an idea of the important and influential words used by each president. The stop words are removed before generating the word cloud.



2.4.1. The word cloud for the 1941-Roosevelt Speech



2.4.2. The word cloud for the 1961-Kennedy Speech



2.4.3. The word cloud for the 1973-Nixon Speech

Insights:

- It is observed that there are a few words (nation, let, us, peace, world) which have good influence and commonly found among all three speeches.
- These words might have helped the presidents to influence the people and create a positive sentiment towards them.
- The word 'nation' is observed in all three speeches and might have significant importance as it creates a sense of unity among the people and makes them move towards a common good / purpose.