MACHINE LEARNING BUSINESS REPORT

March 01'22

PGPDSBA Online Sept_2021

By: Alind Khanna

TABLE OF CONTENTS

PART-1

Executive Summary: Given Data set contains various characteristics of Election data between two electoral candidates. In this problem we need to identify the party favoured by voters on basis of information provided.

Introduction: Purpose of the problem is to design a exit poll, which will help us identifying the wining candidate. Dataset consists of **1525 records** with **9 attributes** determining the electoral candidate.

Data Ingestion:

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

- We need to observe whether data set is correctly uploaded, by using head function we check for first 5 records

	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

- Since unnamed zero is just index of data and we don't need it further during analysis hence we have dropped the same.

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	1	43.0	3.0	3.0	4.0	1.0	2.0	2.0	0.0
1	1	36.0	4.0	4.0	4.0	4.0	5.0	2.0	1.0
2	1	35.0	4.0	4.0	5.0	2.0	3.0	2.0	1.0
3	1	24.0	4.0	2.0	2.0	1.0	4.0	0.0	0.0
4	1	41.0	2.0	2.0	1.0	1.0	6.0	2.0	1.0

- We check for the shape of data (I.e., rows and columns in data set)

```
data_df1.shape
```

```
(1525, 9)
```

- We check for data types and summarize using info function

  There are 7 numeric and 2 categorical variables (Vote & Gender)

```
<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
```

- We check for missing values & duplicates if present

```
data_df1.isnull().sum()
```

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

```
dups=data_df1.duplicated()
print("Total no of duplicate values = %d" % (dups.sum()))
data_df1[dups]
```

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

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

Since we cannot identify uniqueness in duplicate values found hence, we will not remove the same.

- We check for summary of dataset

	count	mean	std	min	25%	50%	75%	max
vote	1525.0	0.697049	0.459685	0.0	0.0	1.0	1.0	1.0
age	1525.0	54.182295	15.711209	24.0	41.0	53.0	67.0	93.0
economic.cond.national	1525.0	3.258033	0.852938	1.5	3.0	3.0	4.0	5.0
economic.cond.household	1525.0	3.161639	0.885286	1.5	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
gender	1525.0	0.467541	0.499109	0.0	0.0	0.0	1.0	1.0

- <u>**Data Skewness**</u>

```
skewness of Unnamed: 0 : -0.8576041066179676
skewness of vote : 0.14447848346551462
skewness of age : -0.2402163142518291
skewness of economic.cond.national : -0.14940490939119963
skewness of economic.cond.household : -0.5348918666133158
skewness of Blair : 0.15194998016716968
skewness of Hague : -0.13581295528712456
skewness of Europe : -0.4264178682034399
skewness of political.knowledge : 0.13011052074203272
```

Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean.

Only two variables are positively skewed and rest negatively skewed with max skewedness in Blair

2) **Perform Univariate and Bivariate Analysis. Do exploratory data analysis.**

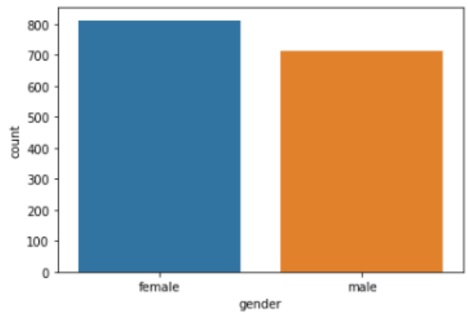
Check for Outliers.

- **<u>Uni+</u>**
- **<u>variate and Bivariate Analysis</u>**

<u>Univariate Analysis</u>

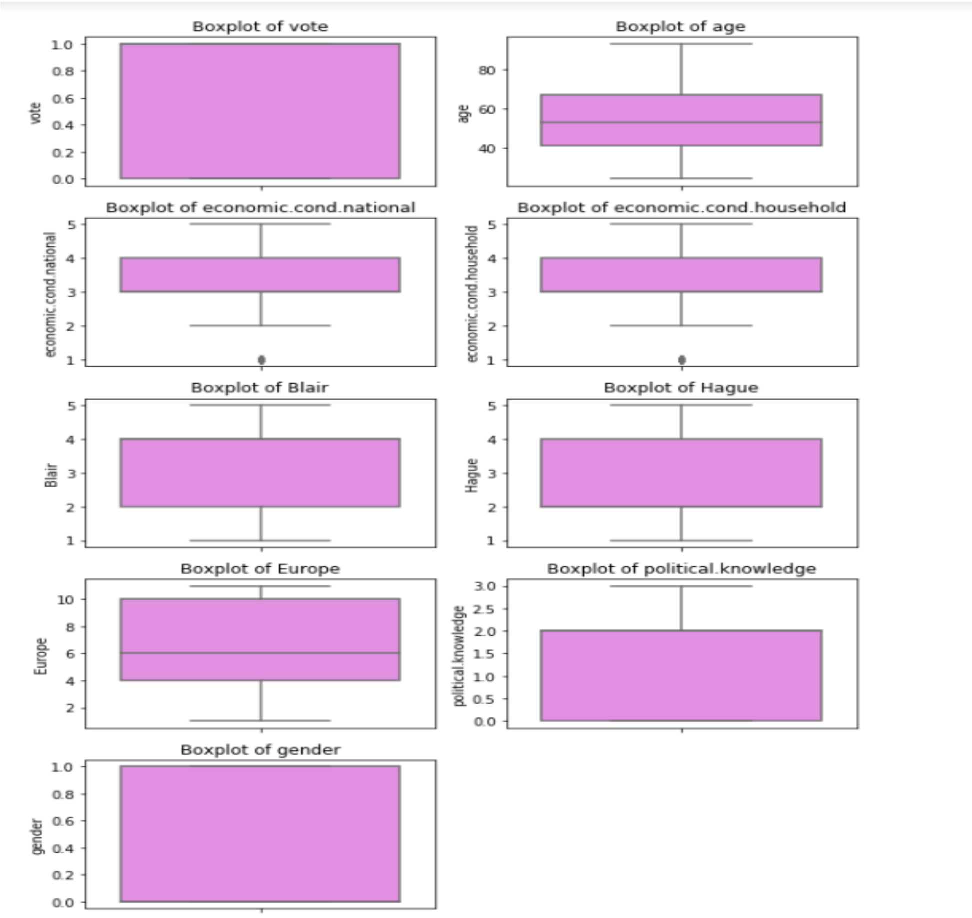
To perform univariate analysis, we have considered **7 continuous variables**, we have used hist plot to analyse the data

Count Plot between gender



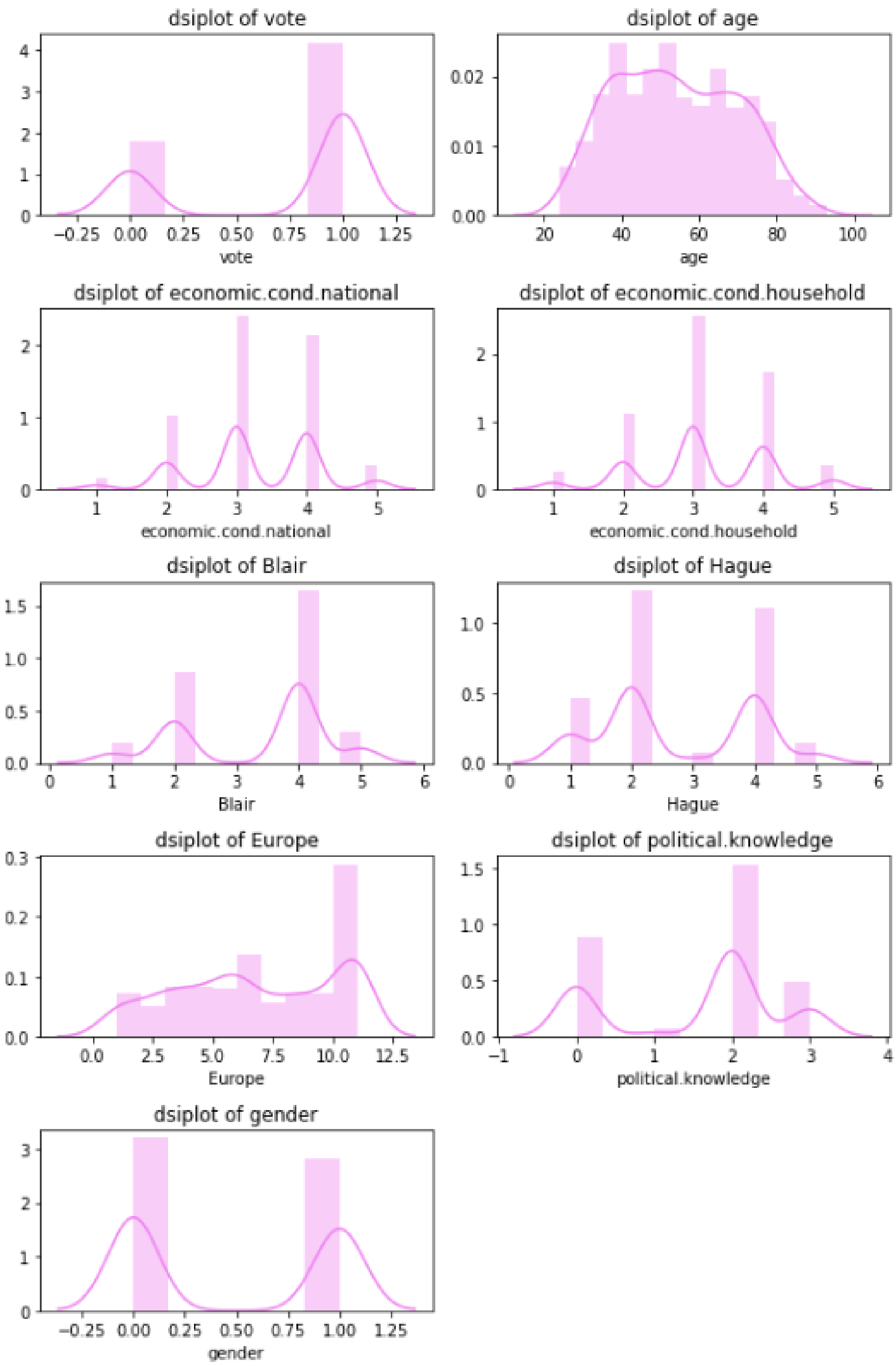
Above plots shows there were maximum of female voters

<u>Box Plots:</u>



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

Dist. Plots:

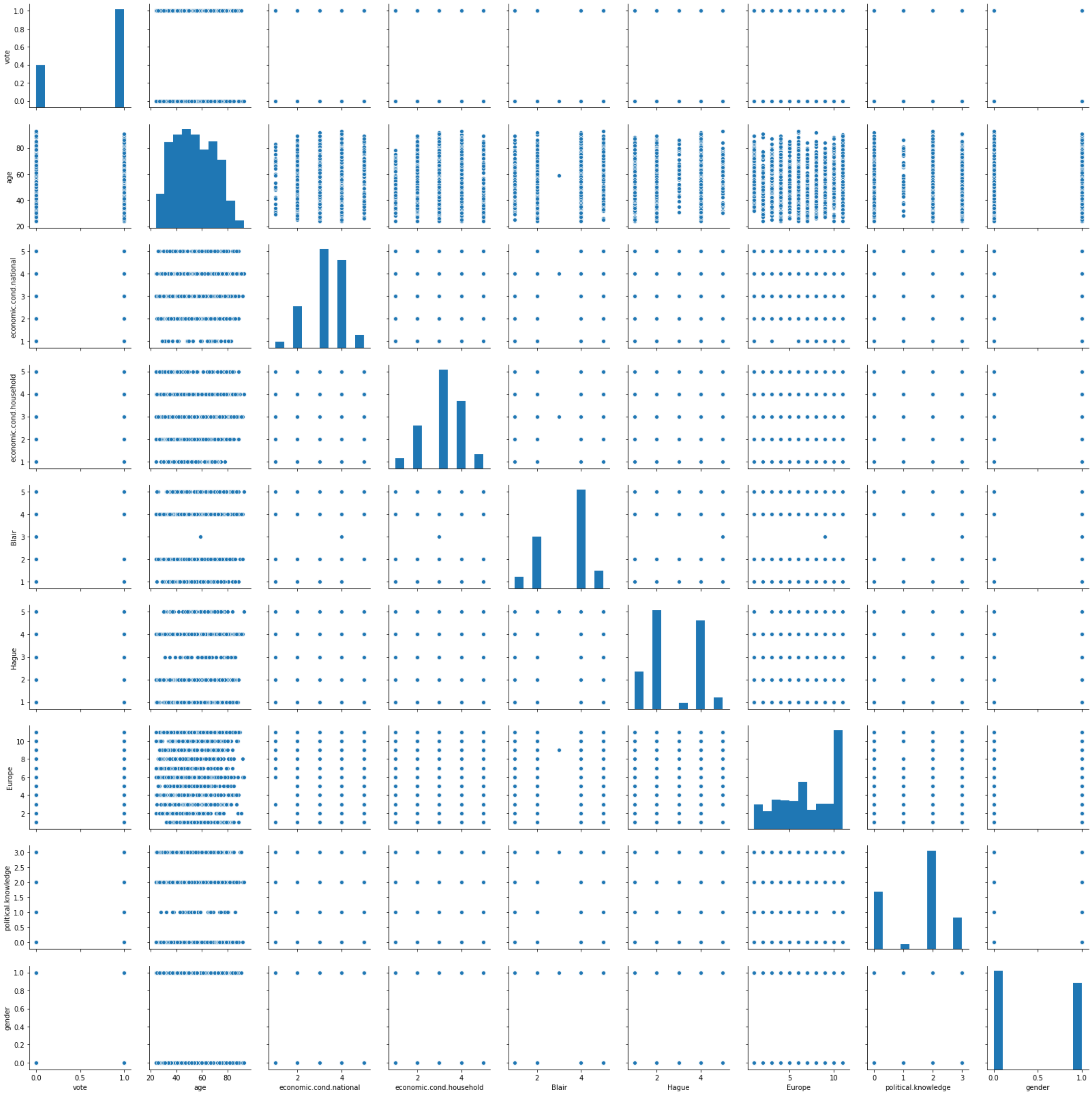


As observed numerical Variables are normally distributed (in some instances are multi modal).

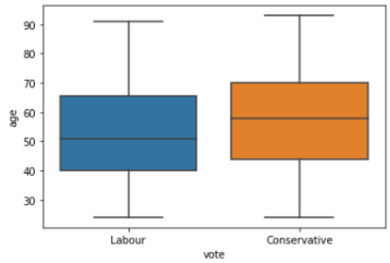
There are outliers present in “economic_cond_national” and “economic_cond_household” variables that can be seen from the boxplots.

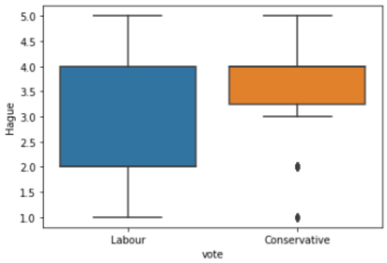
<u>**Bivariate Analysis**</u>

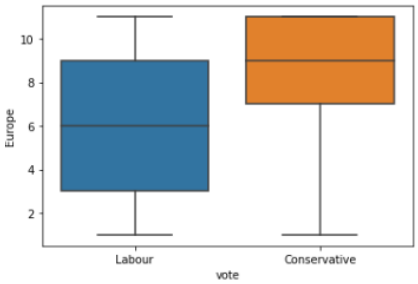
Bivariate analysis is used to understand **interaction between different variables**, this is achieved using **pair plot**.

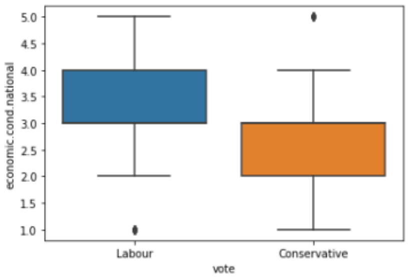


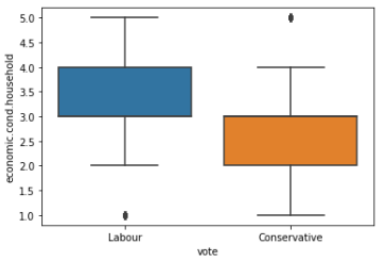
Bivariate Analysis Between Different Variables











Heat Map Correlation

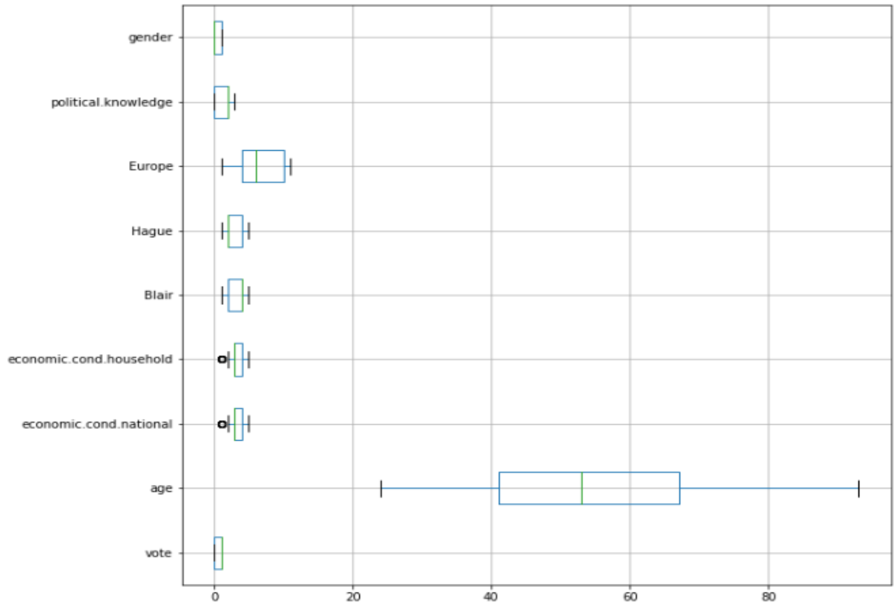
We use heat map to study **correlation between the variables.**

There is strong positive correlation between “economic_cond_national” and “economic_cond_household”

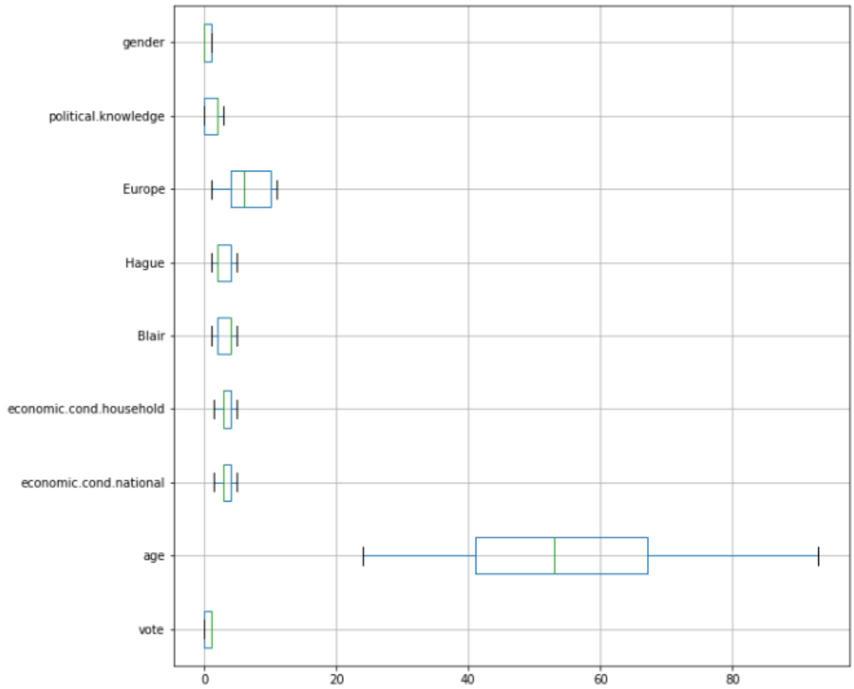
There is strong negative correlation between “blair” and “europe”

Outlier Treatment

Before treatment: We can see there are outliers present in economic. cond. national and economic. condition. Household



After Treatment



Data Preparation:

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)

There are two categorical variables **(Gender & Vote)**, we have used label encoding to encode both of the variables

:

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	1	43.0	3.0	3.0	4.0	1.0	2.0	2.0	0.0
1	1	36.0	4.0	4.0	4.0	4.0	5.0	2.0	1.0
2	1	35.0	4.0	4.0	5.0	2.0	3.0	2.0	1.0
3	1	24.0	4.0	2.0	2.0	1.0	4.0	0.0	0.0
4	1	41.0	2.0	2.0	1.0	1.0	6.0	2.0	1.0

```
data_df1.dtypes
```

```
vote                       int32
age                        int64
economic.cond.national     int64
economic.cond.household    int64
Blair                      int64
Hague                      int64
Europe                     int64
political.knowledge        int64
gender                     int32
dtype: object
```

After encoding next step is to check if there is need for scaling data

Scaling of data is done to align all the variables under similar range, we will use the z score scaling technique to scale the data before data modelling

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	-0.711973	-0.302622	-0.182644	0.566716	-1.419886	-1.434426	0.422643	-0.937059
1	-1.157661	0.870182	0.947305	0.566716	1.018544	-0.524358	0.422643	1.067169
2	-1.221331	0.870182	0.947305	1.418187	-0.607076	-1.131070	0.422643	1.067169
3	-1.921698	0.870182	-1.312594	-1.136225	-1.419886	-0.827714	-1.424148	-0.937059
4	-0.839313	-1.475425	-1.312594	-1.987695	-1.419886	-0.221002	0.422643	1.067169

Data Splitting

Before splitting data, we find target variable, here target variable is 'vote'

Hereafter we split data into test and train, with TEST constituting 30 % and TRAIN 70% respectivel

X_train - denotes 70% training dataset with 8 columns (except the target column called "vote"). X_test- denotes 30% test dataset with 8 columns (except the target column called "vote"). y_train- denotes the 70% training dataset with only the target column called "vote".

y_test- denotes 30% test dataset with only the target column called "vote".

Modelling:

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

Logistic Regression Model

Accuracy

Train: 0.8406

Test: 0.82096

Probability of Train Set

	0	1
0	0.618157	0.381843
1	0.188700	0.811300
2	0.184191	0.815809
3	0.170954	0.829046
4	0.050746	0.949254

Probability of Test Set

	0	1
0	0.921946	0.078054
1	0.690526	0.309474
2	0.346669	0.653331
3	0.488887	0.511113
4	0.158897	0.841103

Classification matrix Train Data

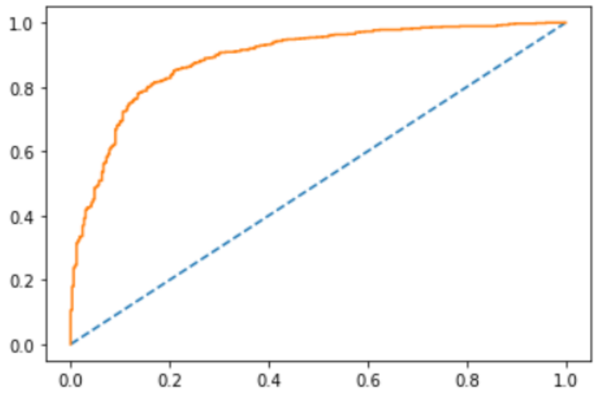
	precision	recall	f1-score	support
0	0.77	0.69	0.73	332
1	0.87	0.91	0.89	735
accuracy			0.84	1067
macro avg	0.82	0.80	0.81	1067
weighted avg	0.84	0.84	0.84	1067

Classification matrix Test Data

	precision	recall	f1-score	support
0	0.70	0.65	0.67	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

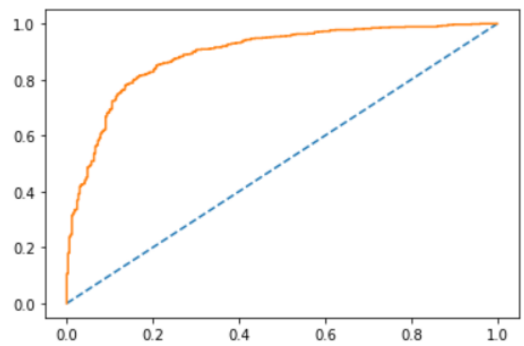
ROC and AUC (Train Data)

AUC: 0.890



ROC and AUC (Test Data)

AUC: 0.883



Linear Discriminant Analysis Model

Accuracy

Train: 0.8397

Test: 0.8187

Classification matrix Train Data

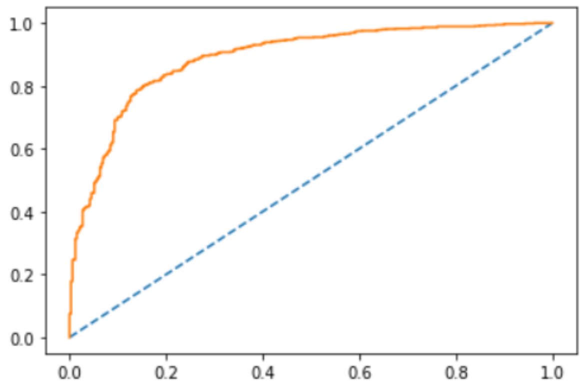
	precision	recall	f1-score	support
0	0.76	0.71	0.73	332
1	0.87	0.90	0.89	735
accuracy			0.84	1067
macro avg	0.82	0.80	0.81	1067
weighted avg	0.84	0.84	0.84	1067

Classification matrix Test Data

	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

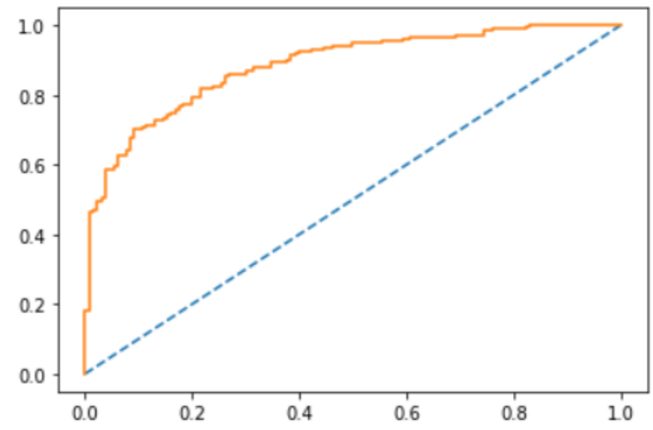
ROC and AUC (Train Data)

the auc 0.889



ROC and AUC (Test Data)

```
the auc curve 0.884
[<matplotlib.lines.Line2D at 0x1c58b965f08>]
```



5) Apply KNN Model and Naïve Bayes Model. Interpret the results.

K NEAREST NEIBHOUR Model

KNN is a distance based supervised machine learning algorithm that can be used to solve both classification and regression problems. Main disadvantage of this model is it becomes very slow when large volume of data

Accuracy

Train: 0.86964

Test: 0.8246

Classification matrix Train Data

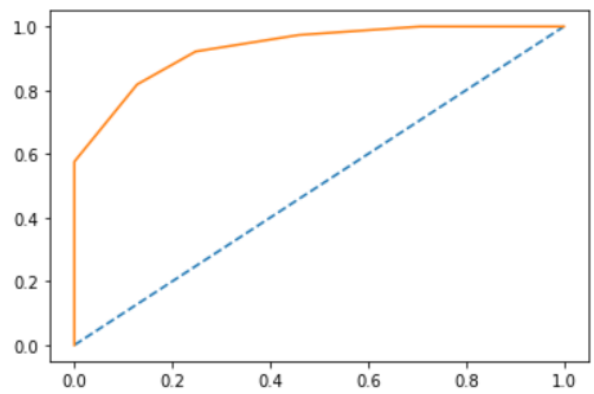
	precision	recall	f1-score	support
0	0.81	0.75	0.78	351
1	0.89	0.92	0.91	792
accuracy			0.87	1143
macro avg	0.85	0.84	0.84	1143
weighted avg	0.87	0.87	0.87	1143

Classification matrix Test Data

	precision	recall	f1-score	support
0	0.69	0.72	0.70	111
1	0.88	0.87	0.88	271
accuracy			0.82	382
macro avg	0.79	0.79	0.79	382
weighted avg	0.83	0.82	0.83	382

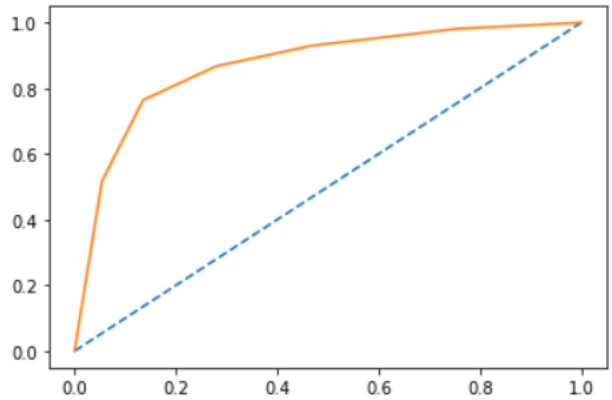
ROC and AUC (Train Data)

```
the auc 0.931
```



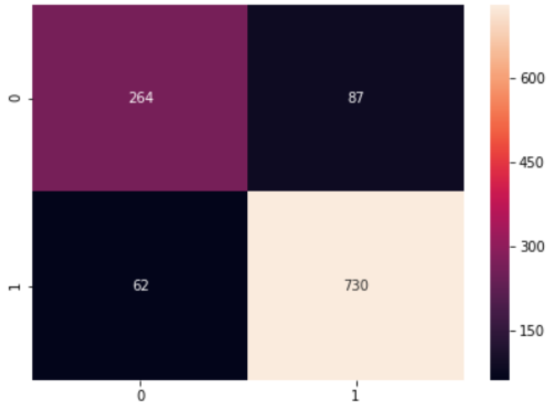
ROC and AUC (Test Data)

```
the auc curve 0.870
[<matplotlib.lines.Line2D at 0x1c58b74a688>]
```



Heat Map for Train data

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c5fcbefd08>
```



FP : 62

FN : 264

TP : 730

TN : 87

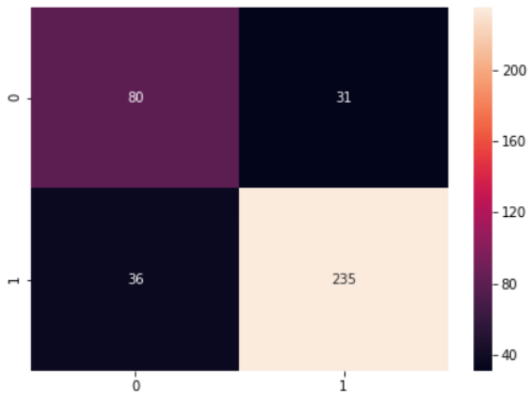
Heat Map for Test data

FP : 36

FN : 80

TP : 235

TN : 31



Heat map shows model responses well in True Positive Cases.

Naiive Bayes Model

Naive Bayes classifiers is a model based on applying Bayes' theorem with strong (naïve) independent assumptions between the features.

Here the method that we are going to use is the GaussianNB() method. This method requires all the features to be in categorical type. A general assumption in this method is the data is following a normal or Gaussian distribution.

Accuracy

Train: 0.83202

Test: 0.82314

Classification matrix Train Data

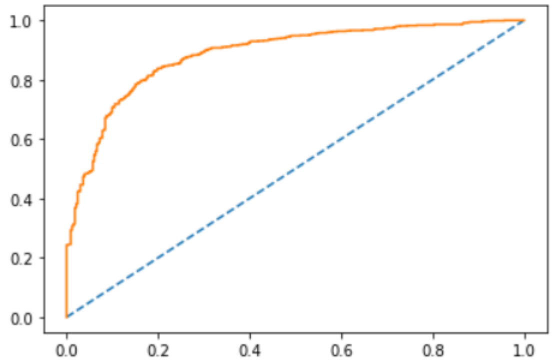
	precision	recall	f1-score	support
0	0.73	0.72	0.72	351
1	0.88	0.88	0.88	792
accuracy			0.83	1143
macro avg	0.80	0.80	0.80	1143
weighted avg	0.83	0.83	0.83	1143

Classification matrix Test Data

	precision	recall	f1-score	support
0	0.68	0.72	0.70	130
1	0.89	0.86	0.87	328
accuracy			0.82	458
macro avg	0.78	0.79	0.79	458
weighted avg	0.83	0.82	0.82	458

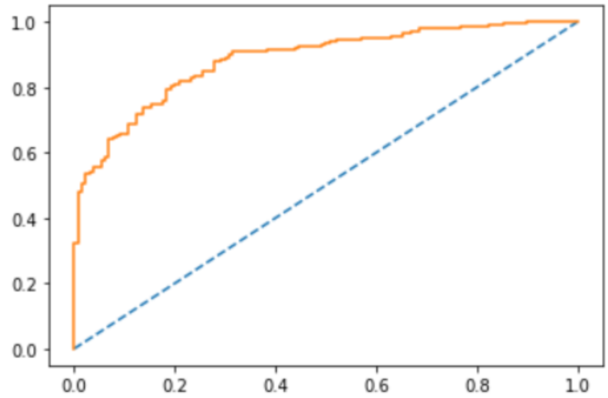
ROC and AUC (Train Data)

```
the auc 0.887
```



ROC and AUC (Test Data)

```
the auc curve 0.885
[<matplotlib.lines.Line2D at 0x1c580bdf808>]
```



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

Model tunning

Model tunning is technique to increase performance of the model without overfitting or creating too high of the variance, this is accomplished by setting hyperparameter.

Overfitting means model will work well on Train data but poor on Test data

Underfitting means model will work well on Test data but poor on Train data

Bagging(Random Forest Classifier)

Bagging is an ensemble technique. Ensemble techniques are the machine learning techniques that combine several base models to get an optimal model. Bagging is designed to improve the performance of existing machine learning algorithms used in statistical classification or regression. It is most commonly used with tree-based algorithms.

Accuracy

Train: 0.96762

Test: 0.8515

Classification matrix Train Data

	precision	recall	f1-score	support
0	0.97	0.92	0.95	351
1	0.97	0.99	0.98	792
accuracy			0.97	1143
macro avg	0.97	0.96	0.96	1143
weighted avg	0.97	0.97	0.97	1143

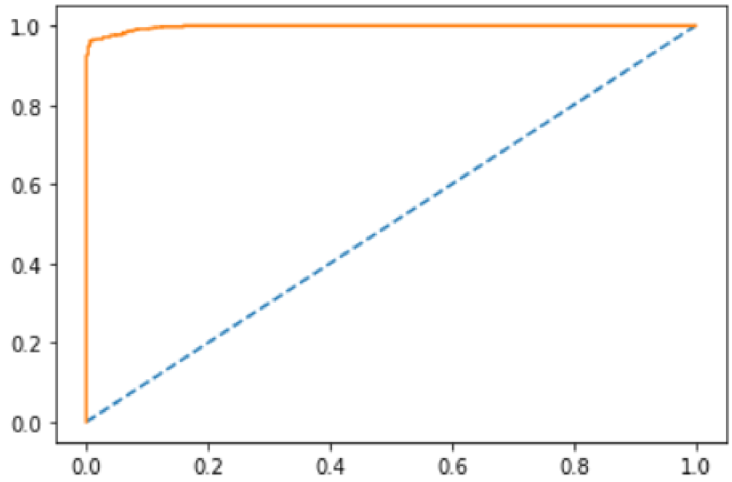
Classification matrix Test Data

	precision	recall	f1-score	support
0	0.74	0.74	0.74	130
1	0.90	0.90	0.90	328
accuracy			0.85	458
macro avg	0.82	0.82	0.82	458
weighted avg	0.85	0.85	0.85	458

ROC and AUC (Train Data)

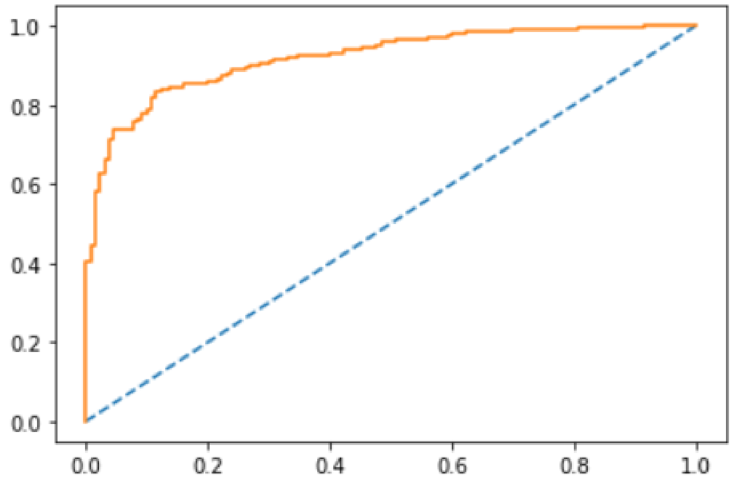
AUC: 0.997

[<matplotlib.lines.Line2D at 0x1c5827b2688>]



ROC and AUC (Test Data)

AUC: 0.919



Boosting Model

Boosting is also an ensemble technique. It converts weak learners to strong learners. Unlike bagging it is a sequential method where result from one weak learner becomes the input for the another and so on, thus improving the performance of the model.

Each time base learning algorithm is applied, it generates a new weak learner prediction rule. This is an iterative process and the boosting algorithm combines these weak rules into a single strong prediction rule.

Misclassified input data gain a higher weight and examples that are classified correctly will lose weight. Thus, future weak learners focus more on the examples that previous weak learners misclassified. They are also tree based methods

ADA Boosting Model

This model is used to increase the efficiency of binary classifiers, but now used to improve multiclass classifiers as well. AdaBoost can be applied on top of any classifier method to learn from its issues and bring about a more accurate model and thus it is called the "best out-of-the-box classifier".

Accuracy

Train: 0.8472

Test: 0.818

Classification matrix Train Data

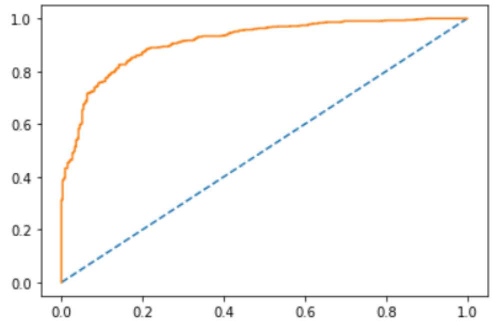
	precision	recall	f1-score	support
0	0.78	0.72	0.74	332
1	0.88	0.91	0.89	735
accuracy			0.85	1067
macro avg	0.83	0.81	0.82	1067
weighted avg	0.84	0.85	0.85	1067

Classification matrix Test Data

	precision	recall	f1-score	support
0	0.68	0.72	0.70	130
1	0.89	0.86	0.87	328
accuracy			0.82	458
macro avg	0.78	0.79	0.79	458
weighted avg	0.83	0.82	0.82	458

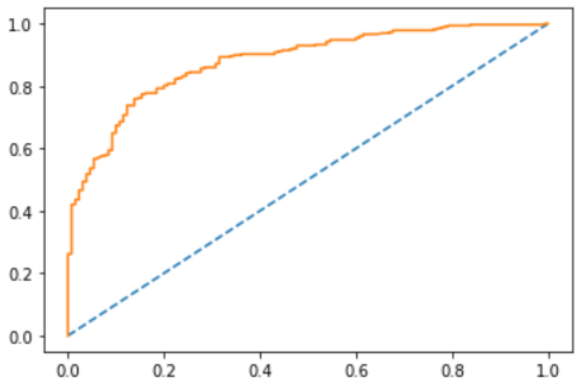
ROC and AUC (Train Data)

```
AUC: 0.913
[<matplotlib.lines.Line2D at 0x1c5828e7c08>]
```



ROC and AUC (Test Data)

```
AUC: 0.879
```



Gradient Based Boosting Model

This model is just like the AD Boosting model. Gradient Boosting works by sequentially adding the misidentified predictors and under-fitted predictions to the ensemble, ensuring the errors identified previously are corrected. The major difference lies in the in what it does with the misidentified value of the previous weak learner. This method tries to fit the new predictor to the residual errors made by the previous one.

Accuracy

Train: 0.886

Test: 0.8318

Classification matrix Train Data

	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

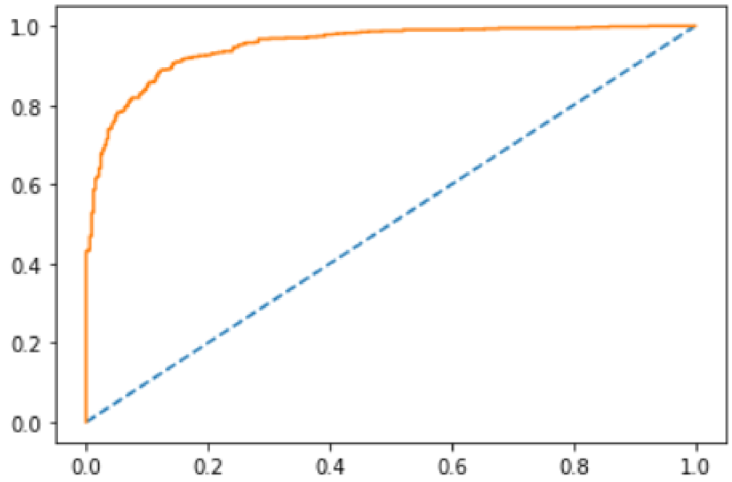
Classification matrix Test Data

	precision	recall	f1-score	support
0	0.68	0.72	0.70	130
1	0.89	0.86	0.87	328
accuracy			0.82	458
macro avg	0.78	0.79	0.79	458
weighted avg	0.83	0.82	0.82	458

ROC and AUC (Train Data)

```
AUC: 0.950
```

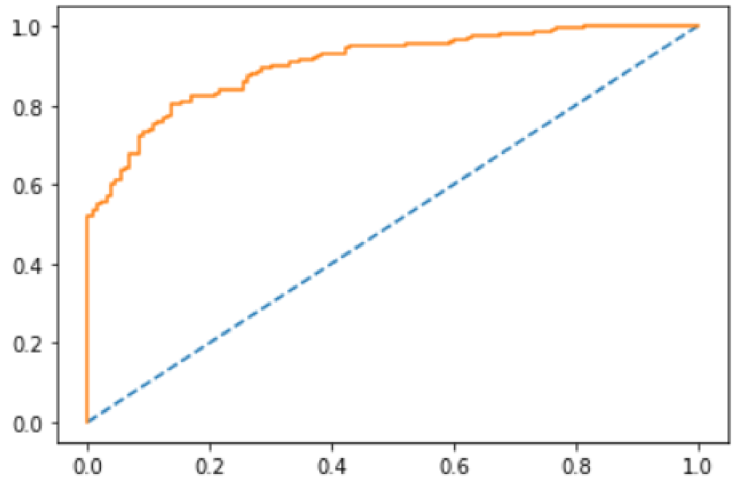
```
[<matplotlib.lines.Line2D at 0x1c5829c3048>]
```



ROC and AUC (Test Data)

```
AUC: 0.904
```

```
[<matplotlib.lines.Line2D at 0x1c5829eecc8>]
```



Model Comparison

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.

	LR(TEST)	LR(TRAIN)	LDA(TEST)	LDA(TRAIN)	KNN(TEST)	KNN(TRAIN)	NB(TEST)	NB(TRAIN)	BAGGING (TEST)	BAGGING(TRAIN)	ADA BOOST(TEST)	ADA BOOST(TRAIN)	GB BOOST(TEST)	GB BOOST(TRAIN)
ACCURACY	0.82	0.84	0.81	0.839	0.82	0.87	0.82	0.83	0.85	0.96	0.82	0.84	0.83	0.89
AUC	0.88	0.89	0.88	0.88	0.87	0.93	0.89	0.89	0.92	0.99	0.88	0.91	0.9	0.95
RECALL	0.89	0.91	0.88	0.9	0.87	0.92	0.86	0.88	0.9	0.99	0.86	0.91	0.86	0.93
PRECISION	0.87	0.87	0.87	0.87	0.88	0.89	0.89	0.88	0.9	0.97	0.89	0.88	0.89	0.91
FI SCORE	0.88	0.89	0.87	0.89	0.88	0.91	0.87	0.88	0.9	0.98	0.87	0.89	0.87	0.92

Here we compare all the models based on there scores and try to find final optimized model

We have statistics of 7 different models run under test and train conditions

Basis of evaluation are

- Accuracy
- AUC
- Recall
- Precision
- F1 Score

From the above data we observe

- Accuracy : Based on accuracy GB Boost performed better
- AUC : Based on AUC score GB Boost performed better
- Recall : Based on recall score LR model performed slightly better
- Precision : Based on precision GB Boost performed better
- F1 Score : Based on precision KNN performed better

Based on given statistics GB Boosting performed much better in all the score quadrants.

8) Based on these predictions, what are the insights?

Our major motive here is to device the model which will give good prediction for which paty voter will vote on the basis of given information. This will help us in creation of exit poll for news channel.

- Using Bagging Random Forest Based Model with scaling for predicting the outcome as it has the best optimised performance
- Gathering more data will also help in training the models and thus improving their predictive powers
- Boosting Models can also perform well like GBBoost performed well even without tuning. Thus, if we perform hyper-parameters tuning, we might get better results

PART-2

Problem 2:

Executive Summary: In the given problem we need to work on speeches of three different USA presidents.

Introduction: There are 3 different speeches of three different speeches corresponding to three different presidents, we need to derive inferences base don problem sets.

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

```
 Roosevelt Speech Characterstics

Number Of Characters 7571
Number Of Words 1536
Number Of Sentences 68

 Kennedy Speech Characterstics

Number Of Characters 7618
Number Of Words 1546
Number Of Sentences 52

 Nixon Speech Characterstics

Number Of Characters 9991
Number Of Words 2028
Number Of Sentences 69
```

2) Remove all the stop words from all three speeches

To remove the stopwords, there is package called "stopwords" in the nltk.corpus library. So, in order to do so we need to import following libraries

- from nltk.corpus import stopwords

- from nltk. stem.porter import PorterStemmer

The stopwords library contains all the stop words like ‘and’, ‘a’, ‘is’, ‘to’, ‘is’, ‘.’, ‘of’, ‘to’ etc., that usually don’t have any importance in understanding the sentiment or usefulness in machine learning algorithms

Stemming is a process which helps the processor in understanding the words that have similar meaning. In this the words are brought down to their base or root level by removing the affixes. It is highly used in search engines. For e.g. - eating, eats, eaten all these will be reduced to eat after stemming

```
Number of Character without stopwords in Roosevelt speech :  4905
Number of Character without stopwords in Kennedy speech :  5057
Number of Character without stopwords in Nixon speech :  6266
Number of Words without stopwords in Roosevelt speech :  666
Number of Words without stopwords in Kennedy speech :  730
Number of Words without stopwords in Nixon speech :  861
```

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)

For 1941-Roosevelt.txt

```
["'s", 'achiev', 'across', 'act', 'action']
```

For 1973-Nixon.txt

```
["'s", '200th', 'abl', 'abov', 'abroad']
```

For 1961-Kennedy.txt

```
["'s", 'abolish', 'absolut', 'accid', 'across']
```

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