Projects: Machine Learning -1&2
By Ganesh Aryan

Problem 1:

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

DATA DICTIONARY:

1. **vote**: Party choice: Conservative or Labour

2. **age**: in years

3. **economic.cond.national**: Assessment of current national economic conditions, 1 to 5.

4. **economic.cond.household**: Assessment of current household economic conditions, 1 to 5.

5. **Blair**: Assessment of the Labour leader, 1 to 5.

6. **Hague**: Assessment of the Conservative leader, 1 to 5.

7. **Europe**: an 11-point scale that measures respondents' attitudes toward European integration. High scores represent 'Eurosceptic' sentiment.

8. **political.knowledge**: Knowledge of parties' positions on European integration, 0 to 3.

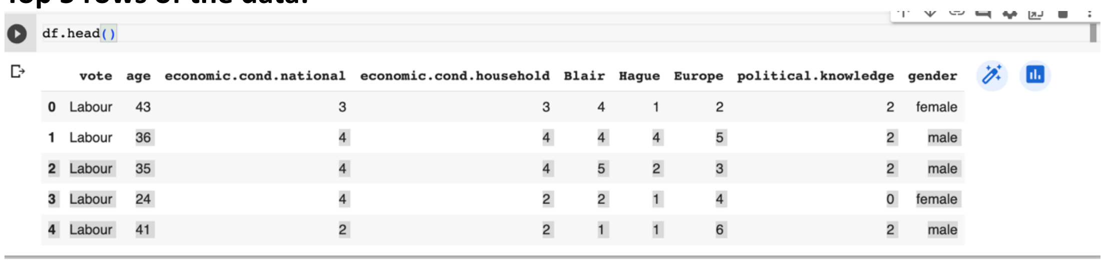
9. **gender**: female or male.

Data Ingestion:

1.1 Read the dataset. Do the descriptive statistics and do the null value condition check. Write an inference on it.
Loading the data from the drive:

```
df = pd.read_excel('Election_Data.xlsx')
```

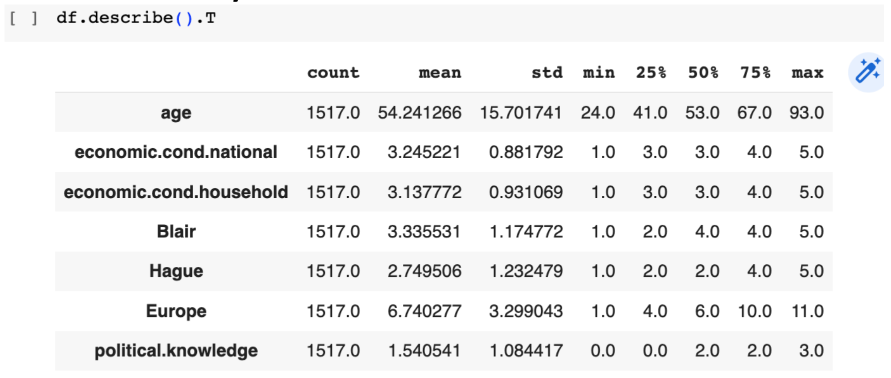
Top 5 rows of the data:



```
df.head()
```

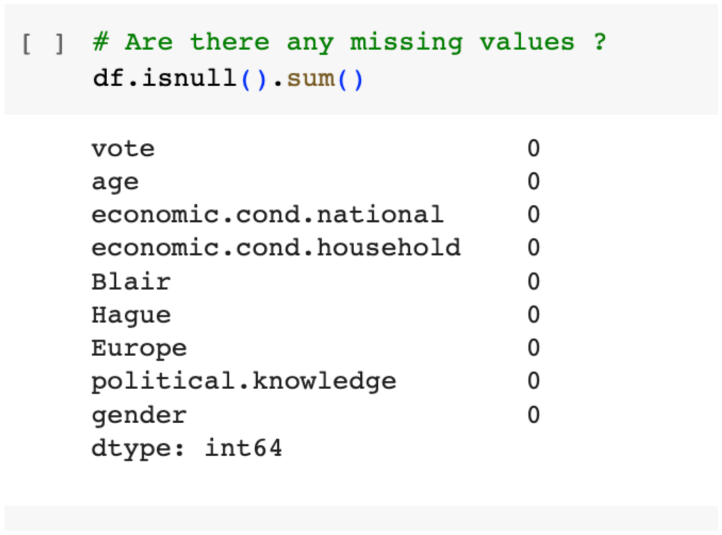
	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	Labour	43	3	3	4	1	2	2	female
1	Labour	36	4	4	4	4	5	2	male
2	Labour	35	4	4	5	2	3	2	male
3	Labour	24	4	2	2	1	4	0	female
4	Labour	41	2	2	1	1	6	2	male

5 Points Summary:



```
[ ] df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
age	1517.0	54.241266	15.701741	24.0	41.0	53.0	67.0	93.0
economic.cond.national	1517.0	3.245221	0.881792	1.0	3.0	3.0	4.0	5.0
economic.cond.household	1517.0	3.137772	0.931069	1.0	3.0	3.0	4.0	5.0
Blair	1517.0	3.335531	1.174772	1.0	2.0	4.0	4.0	5.0
Hague	1517.0	2.749506	1.232479	1.0	2.0	2.0	4.0	5.0
Europe	1517.0	6.740277	3.299043	1.0	4.0	6.0	10.0	11.0
political.knowledge	1517.0	1.540541	1.084417	0.0	0.0	2.0	2.0	3.0



```
[ ] # Are there any missing values ?
    df.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
```

- Data has 1525 rows and 9 columns in it.
- there are 8 numerical and 2 object dtype in it.
- No null values present.
- Data looks skewed a bit towards the left and right.
- Target column is Vote

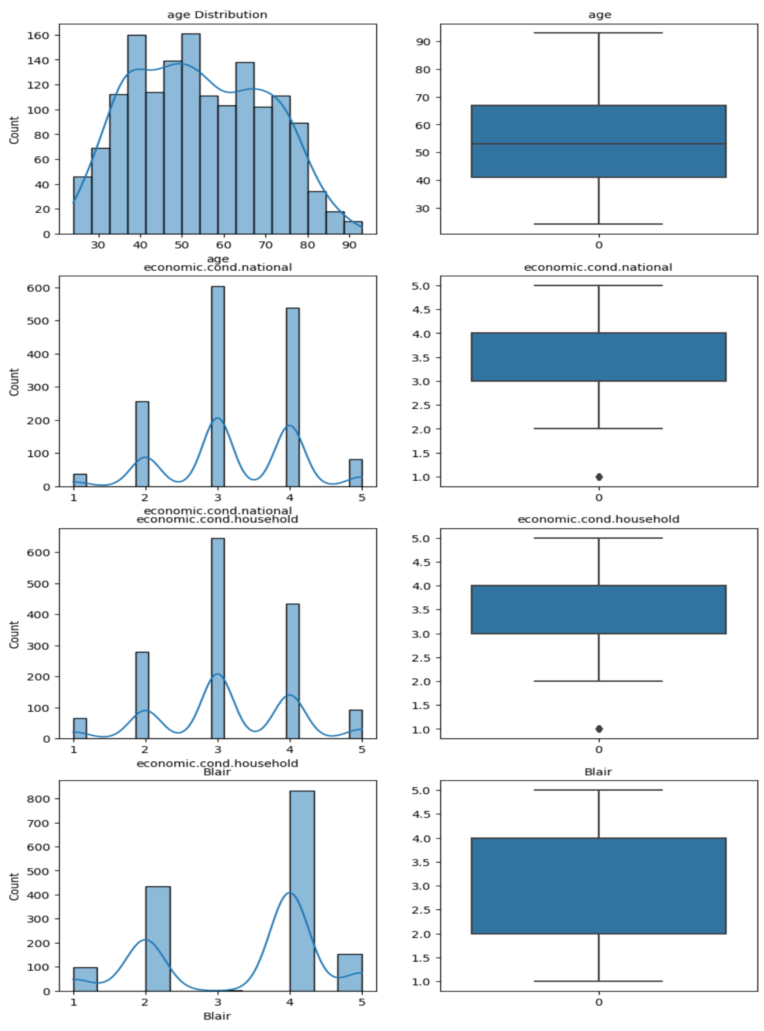
1.2 Perform Univariate and Bivariate Analysis. Do exploratory data analysis. Check for Outliers.

EDA

- There are 8 duplicate values present and same needs to be deleted.
- There seems to be no missing data.
- Data is skewed slight towards both sides.

Univariate Analysis

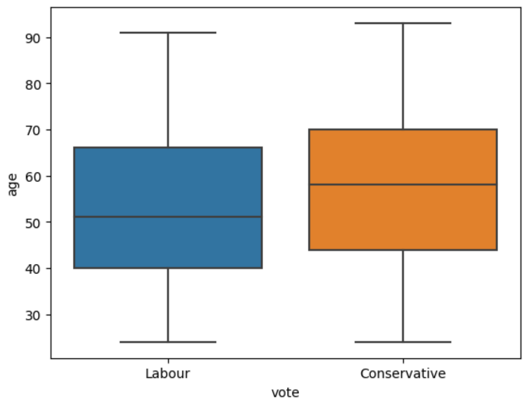
- As per below plots, it is quite evident that most of the voters are between the age group of 40-65
- Voters, current economic conditions both national and Household levels seems to having fairly good assessment.
- As per the below analysis it seems assessment on labour leader is little better as compared to consecutive leader.
- Majority of the respondents seems to having high Eurosceptic sentiments, they will favour the one support disengagement like "Brixit" with EU countries.
- Respondents have fairly good knowledge of party's positions on European integration.

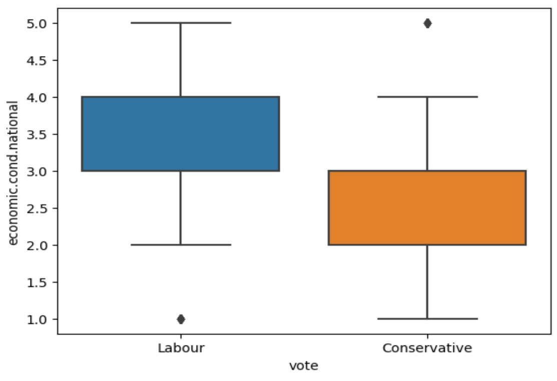


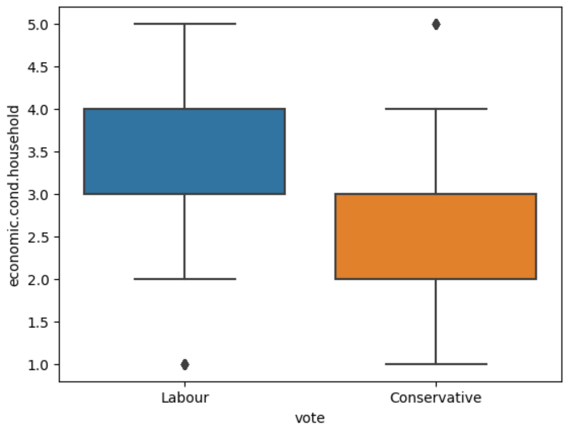
Bivariate Analysis

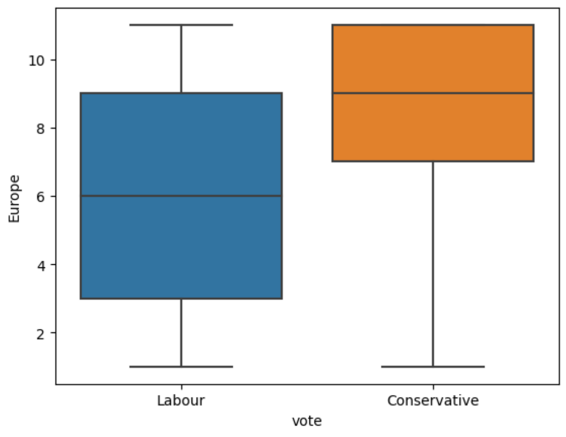
Bivariate analysis is the simultaneous analysis of the variables. It explores the concept of relationships between the two, whether there exists an association and the strength of this association, or whether there are differences between and the significance of the same. On performing the bivariate analysis between “Vote” and “Age” variables we can notice the young population have less probability to vote Conservative as well the old aged.

Voters having higher Assessment of current national economic and Household conditions likely to vote for labour party.



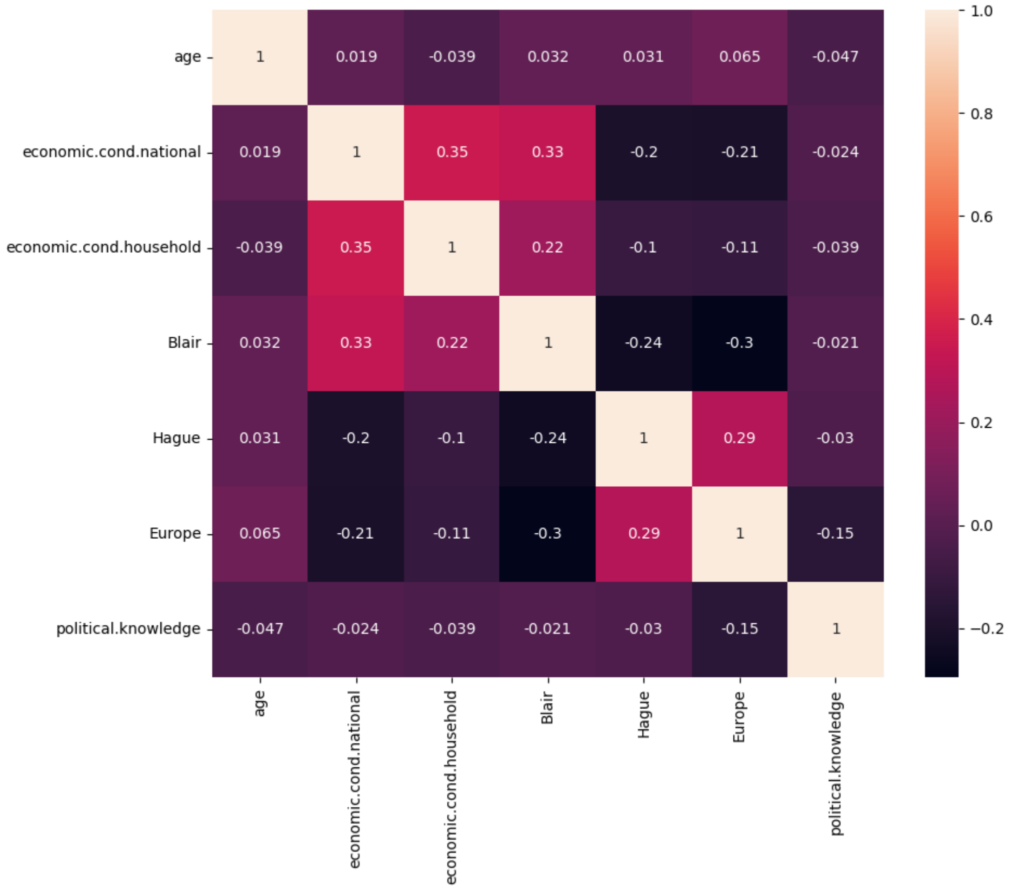






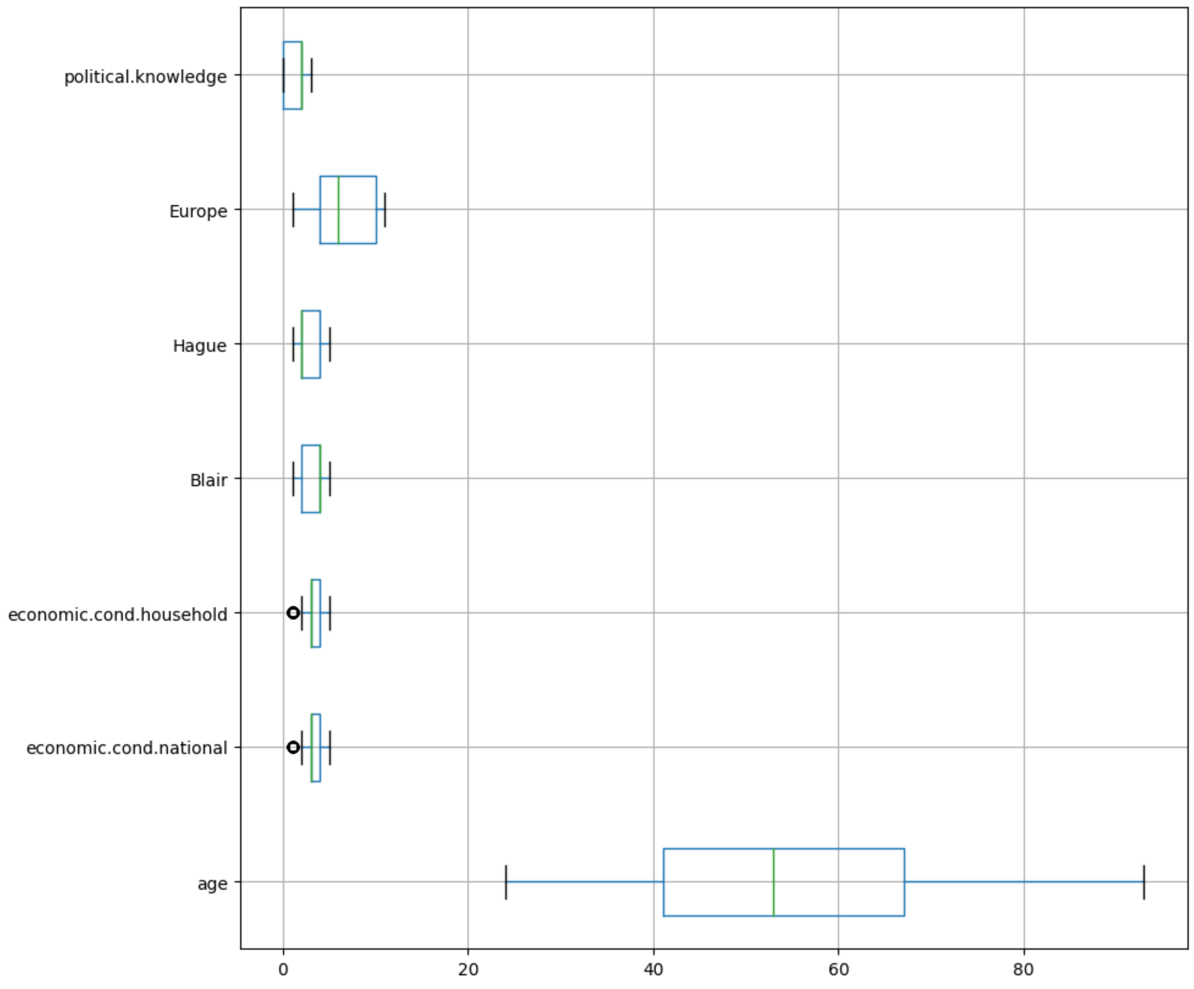
Checking for Corelation between the variables

Heat map shows no significant corelation between variables



	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge
age	1	0.019	-0.039	0.032	0.031	0.065	-0.047
economic.cond.national	0.019	1	0.35	0.33	-0.2	-0.21	-0.024
economic.cond.household	-0.039	0.35	1	0.22	-0.1	-0.11	-0.039
Blair	0.032	0.33	0.22	1	-0.24	-0.3	-0.021
Hague	0.031	-0.2	-0.1	-0.24	1	0.29	-0.03
Europe	0.065	-0.21	-0.11	-0.3	0.29	1	-0.15
political.knowledge	-0.047	-0.024	-0.039	-0.021	-0.03	-0.15	1

Checking for outliers



No Outliers present in to the DataF.

Data Preparation:

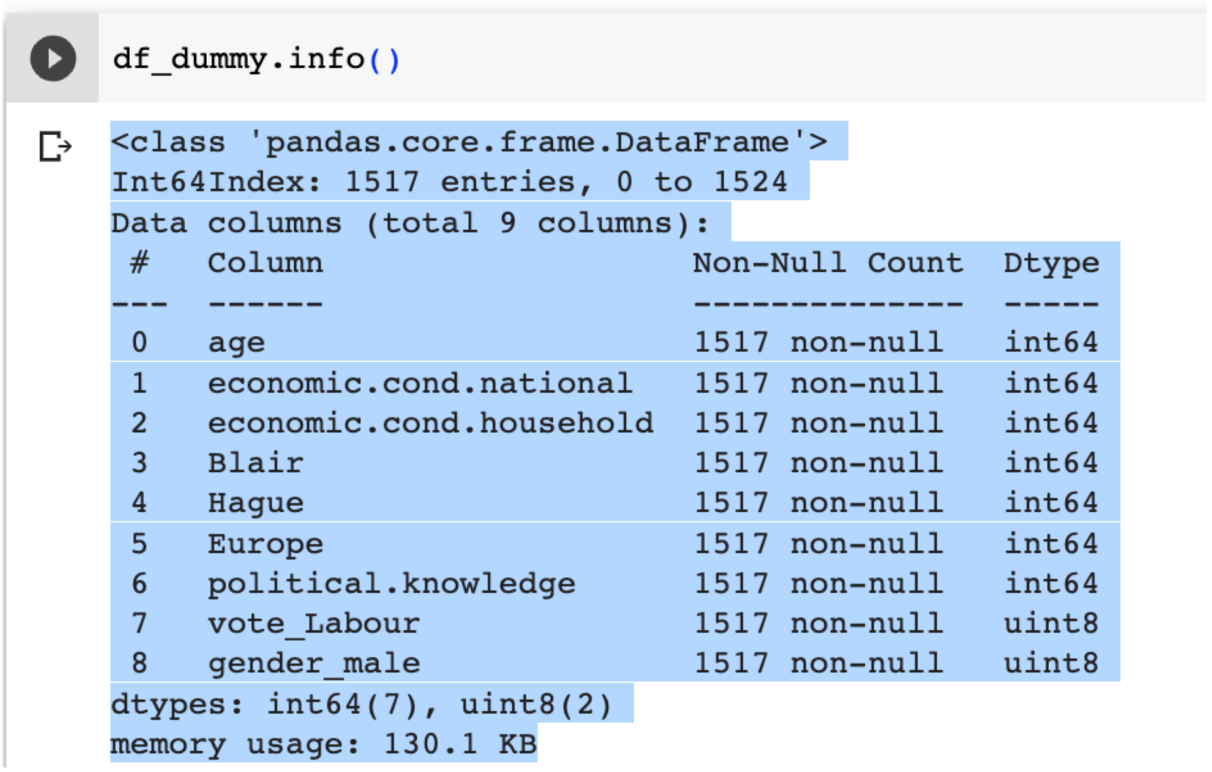
1.3 Encode the data (having string values) for Modelling. Is Scaling necessary here or not? Data Split: Split the data into train and test (70:30).

Encoding:
Converting the other 'object' type variables as dummy variables

```
df_dummy = pd.get_dummies(df,drop_first=True)
```

After:

```
df_dummy.info()
```



```
df_dummy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1517 entries, 0 to 1524
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   age                      1517 non-null   int64
 1   economic.cond.national   1517 non-null   int64
 2   economic.cond.household  1517 non-null   int64
 3   Blair                    1517 non-null   int64
 4   Hague                    1517 non-null   int64
 5   Europe                   1517 non-null   int64
 6   political.knowledge      1517 non-null   int64
 7   vote_Labour              1517 non-null   uint8
 8   gender_male              1517 non-null   uint8
dtypes: int64(7), uint8(2)
memory usage: 130.1 KB
```

Checking data for scaling

It observed that data scaling isn’t necessary for our model building

Split the data into train and test (70:30).

```
# capture the target column ("default") into separate vectors for
training set and test set
#X = df.drop("vote" , axis=1)
#y = df.pop("vote")
# Copy all the predictor variables into X dataframe
X = df_dummy.drop('vote_Labour', axis=1)
y = df_dummy['vote_Labour']
```

Train & test Split

```
# Copy all the predictor variables into X dataframe
X = df_dummy.drop('vote_Labour', axis=1)

# Copy target into the y dataframe.
y = df_dummy['vote_Labour']
# Split X and y into training and test set in 70:30 ratio
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.30 , random_state=1,stratify=df_dummy['vote_Labour'])
```

- As instructed the data has been split in 70:30 ratio.
- Made a split of target variable as Y and rest as X.
- Vote_labour is our target class as 1

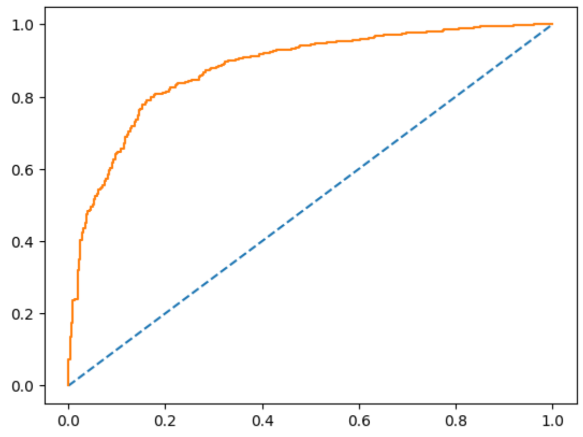
Modelling:

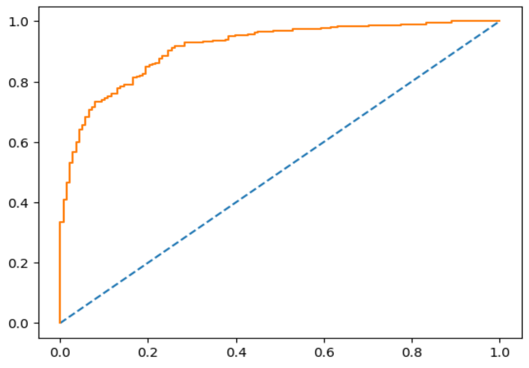
1.4 Apply Logistic Regression and LDA (linear discriminant analysis).

Logistic Regression

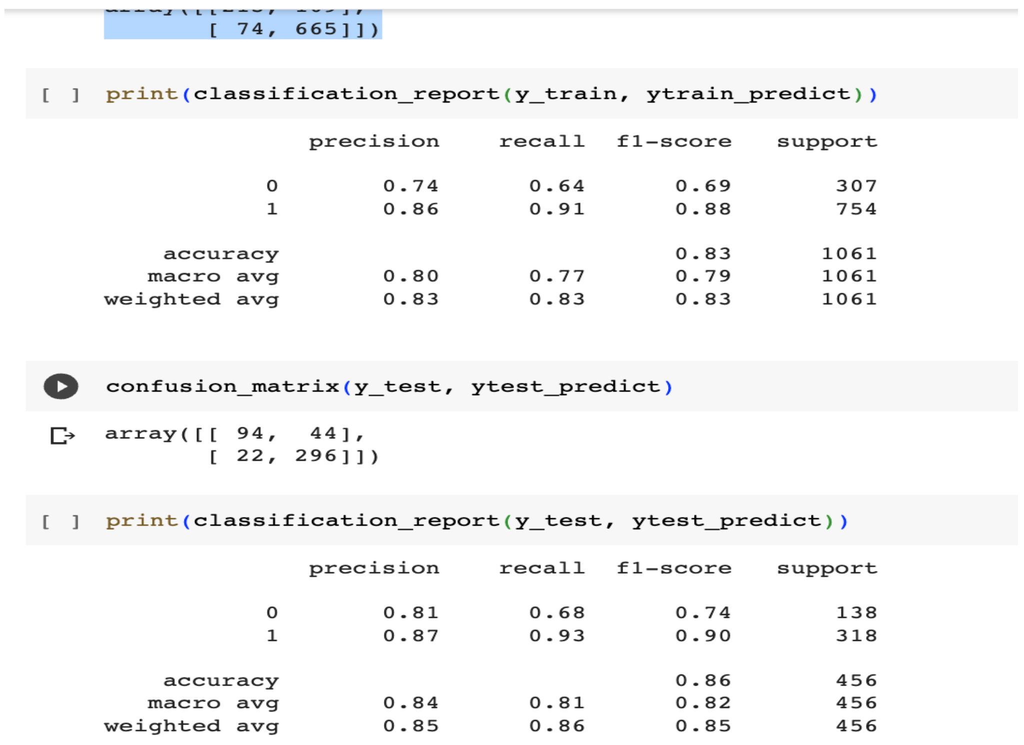
Model predicted

- Model Score has predicted 83% train and 86% on test data.
- AUC towards 1 indicates that model has performed well here.
- For predicting to vote (Label 0)to Conservative :
- Precision (81%) – 81% of the population will vote for conservatives.
- Recall (68%) – Out of all population will vote for conservatives.
- For predicting to Vote Labour (Label 1):
- Precision (87%) – 87% of the population will vote for Labour.
- Recall (87%) – 87% Out of all population will vote for Labour.





Confusion Matrix



```
              [ 74, 665]])
```

```
[ ] print(classification_report(y_train, ytrain_predict))
```

```
              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
```

```
confusion_matrix(y_test, ytest_predict)
```

```
array([[ 94,  44],
       [ 22, 296]])
```

```
[ ] print(classification_report(y_test, ytest_predict))
```

```
              precision    recall  f1-score   support

           0       0.81      0.68      0.74       138
           1       0.87      0.93      0.90       318

    accuracy                           0.86       456
   macro avg       0.84      0.81      0.82       456
weighted avg       0.85      0.86      0.85       456
```

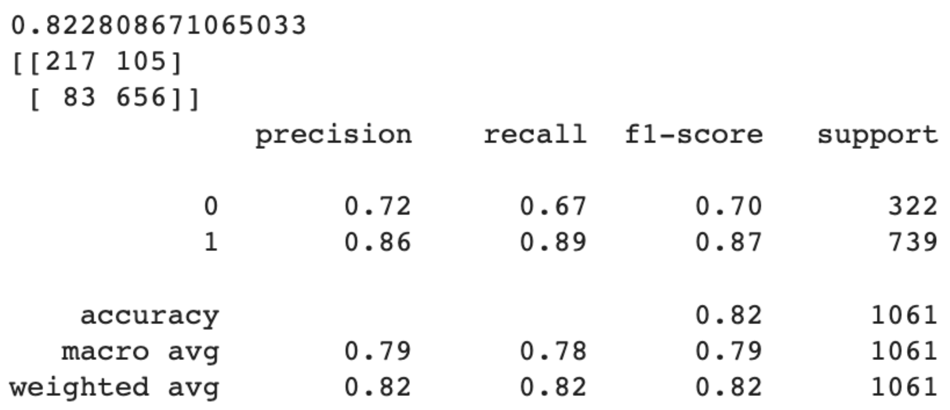
LDA (linear discriminant analysis)

Building a LDA model and use this model to predict on the test set and compute the confusion matrix.

Model Evaluation

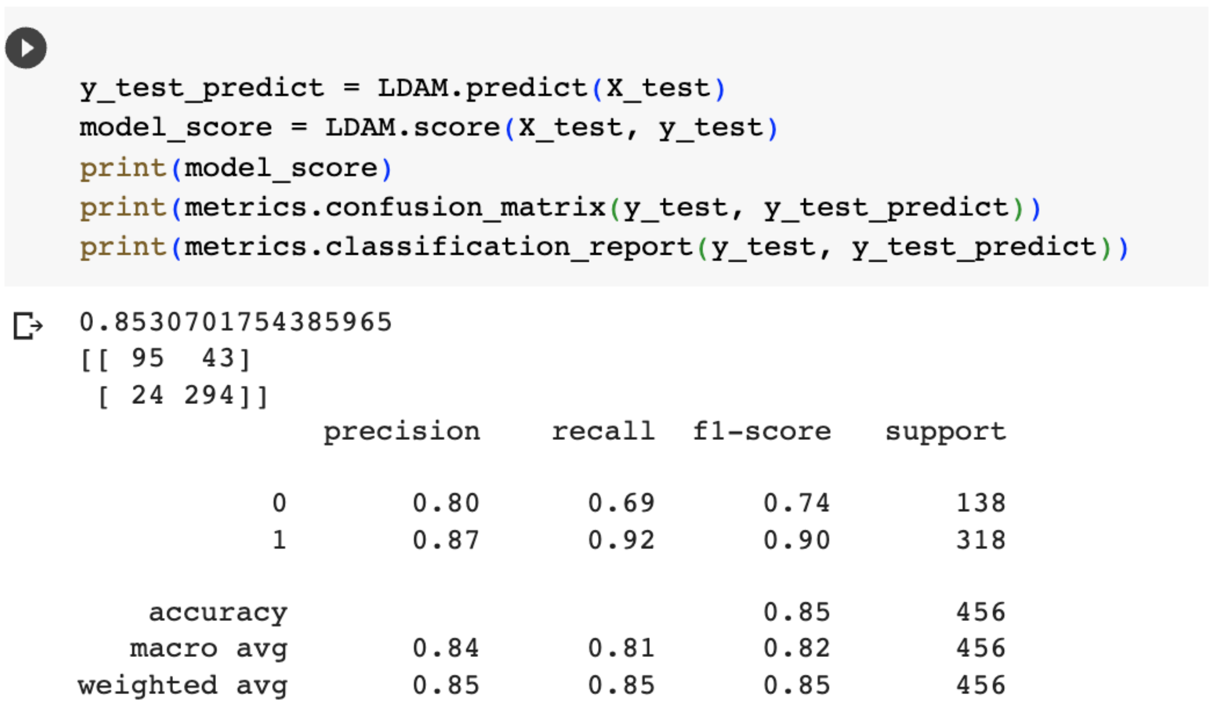
- 82% accuracy for training data.
- 85% accuracy on test data.
- AUC score 87% & 89% for both train and test indicate that model has perform quite good in both the occasions.

Performance Matrix on train Data



```
0.822808671065033
[[217 105]
 [ 83 656]]
              precision    recall  f1-score   support

           0       0.72      0.67      0.70       322
           1       0.86      0.89      0.87       739

    accuracy                           0.82      1061
   macro avg       0.79      0.78      0.79      1061
weighted avg       0.82      0.82      0.82      1061
```

Performance Matrix on test data set

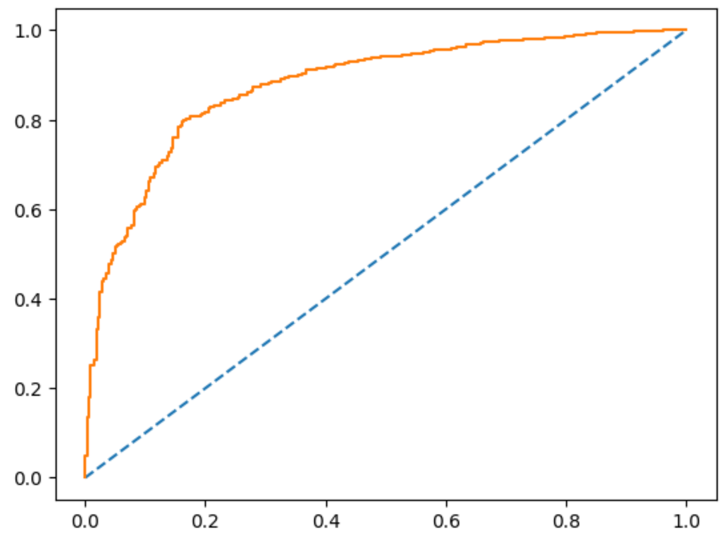


```
y_test_predict = LDAM.predict(X_test)
model_score = LDAM.score(X_test, y_test)
print(model_score)
print(metrics.confusion_matrix(y_test, y_test_predict))
print(metrics.classification_report(y_test, y_test_predict))
```

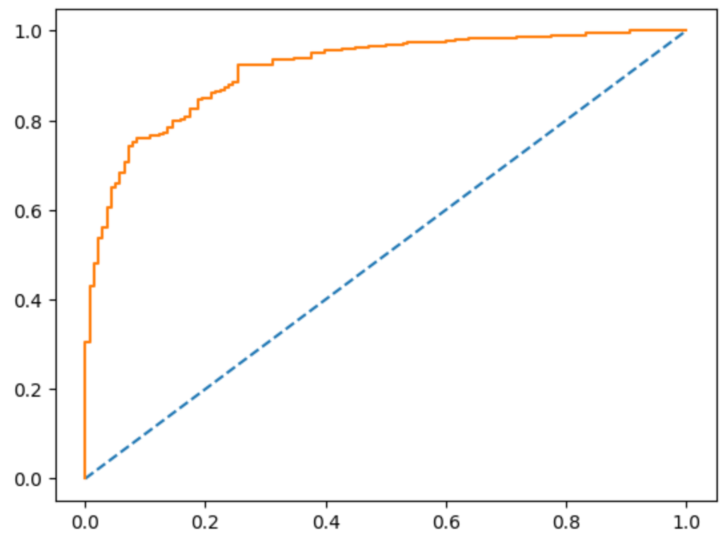
```
0.8530701754385965
[[ 95  43]
 [ 24 294]]
              precision    recall  f1-score   support

           0       0.80      0.69      0.74       138
           1       0.87      0.92      0.90       318

    accuracy                           0.85       456
   macro avg       0.84      0.81      0.82       456
weighted avg       0.85      0.85      0.85       456
```

AUC Score:

0.87 on train:



0.91 on test data:



1.5 Apply KNN Model and Naïve Bayes Model. Interpret the results.

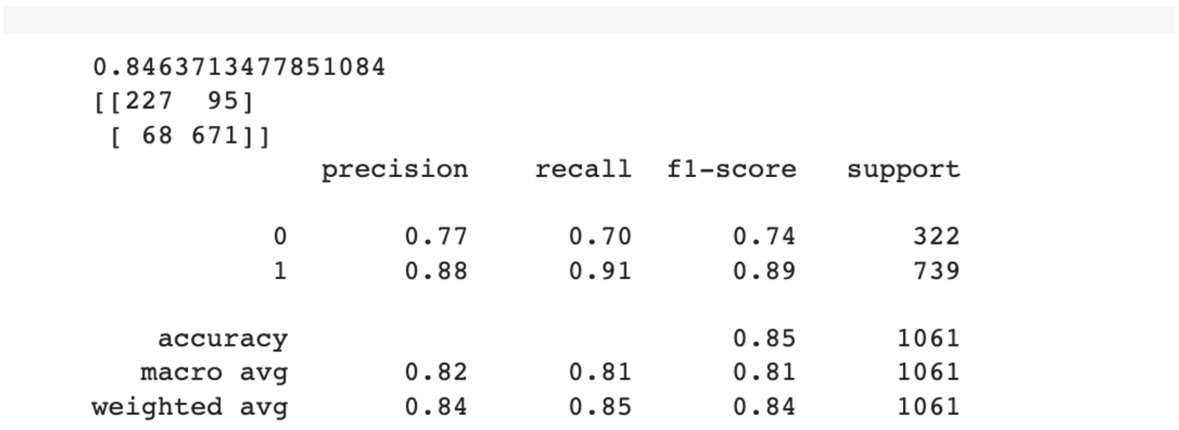
KNN Model:

Here we have observed that with KNN both train and test data has given 85% and 80% respectively accuracy.

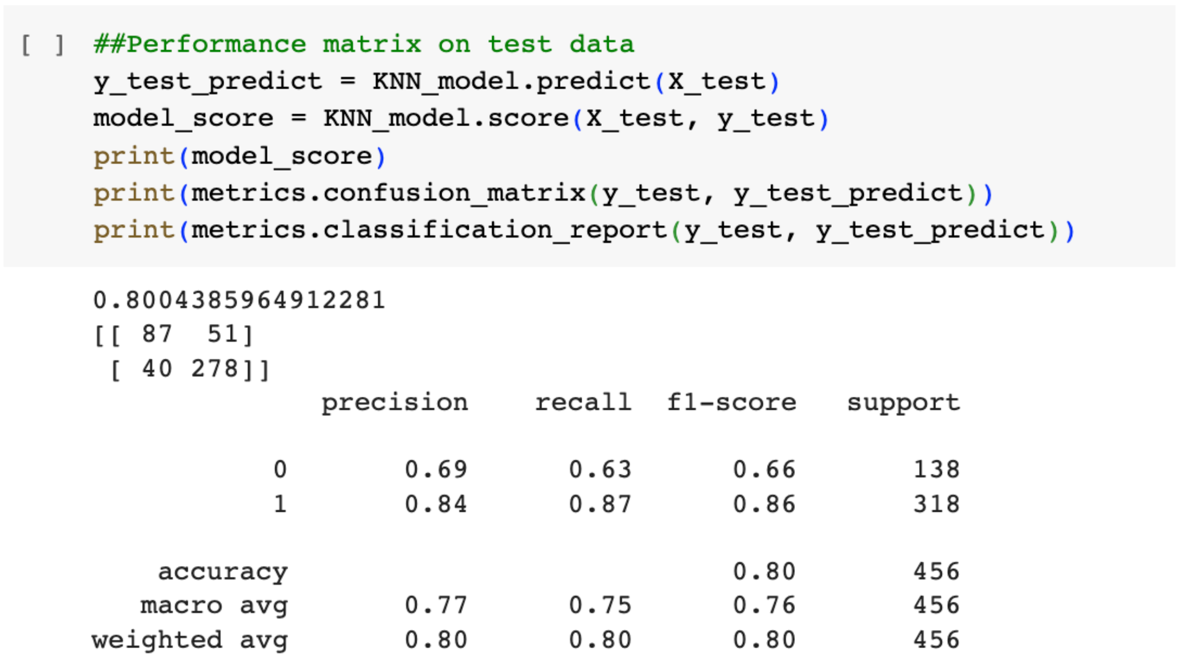
It has predicted similar range of previous models.

AUC score for train 0.91

AUC score for test 0.85



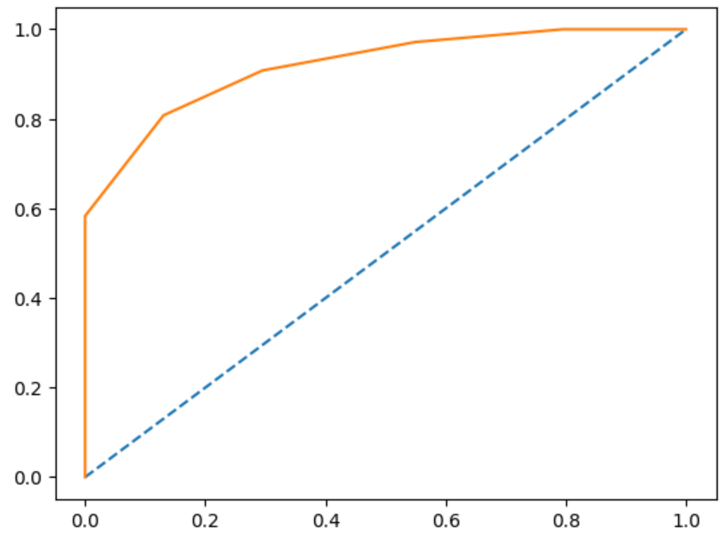
```
0.8463713477851084
[[227  95]
 [ 68 671]]
              precision    recall  f1-score   support

           0       0.77      0.70      0.74       322
           1       0.88      0.91      0.89       739

    accuracy                           0.85      1061
   macro avg       0.82      0.81      0.81      1061
weighted avg       0.84      0.85      0.84      1061
```



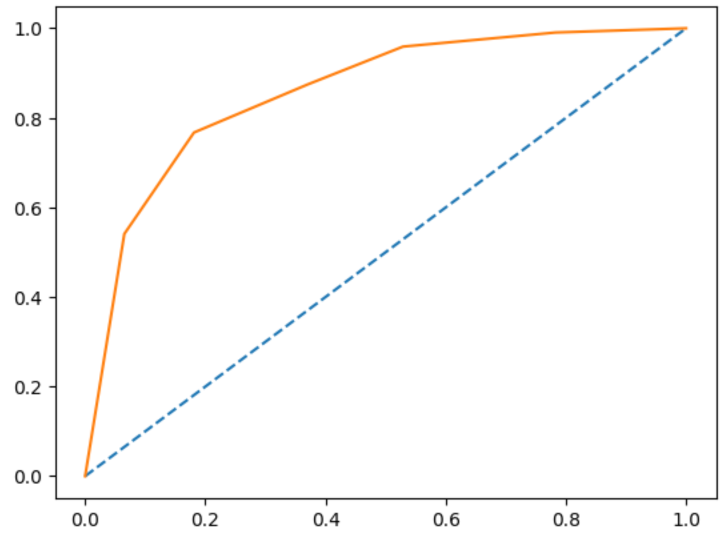
```
[ ] ##Performance matrix on test data
    y_test_predict = KNN_model.predict(X_test)
    model_score = KNN_model.score(X_test, y_test)
    print(model_score)
    print(metrics.confusion_matrix(y_test, y_test_predict))
    print(metrics.classification_report(y_test, y_test_predict))
```

```
0.8004385964912281
[[ 87  51]
 [ 40 278]]
              precision    recall  f1-score   support

           0       0.69      0.63      0.66       138
           1       0.84      0.87      0.86       318

    accuracy                           0.80       456
   macro avg       0.77      0.75      0.76       456
weighted avg       0.80      0.80      0.80       456
```

AUC- Train



AUC- Test



Naïve Bayes Model

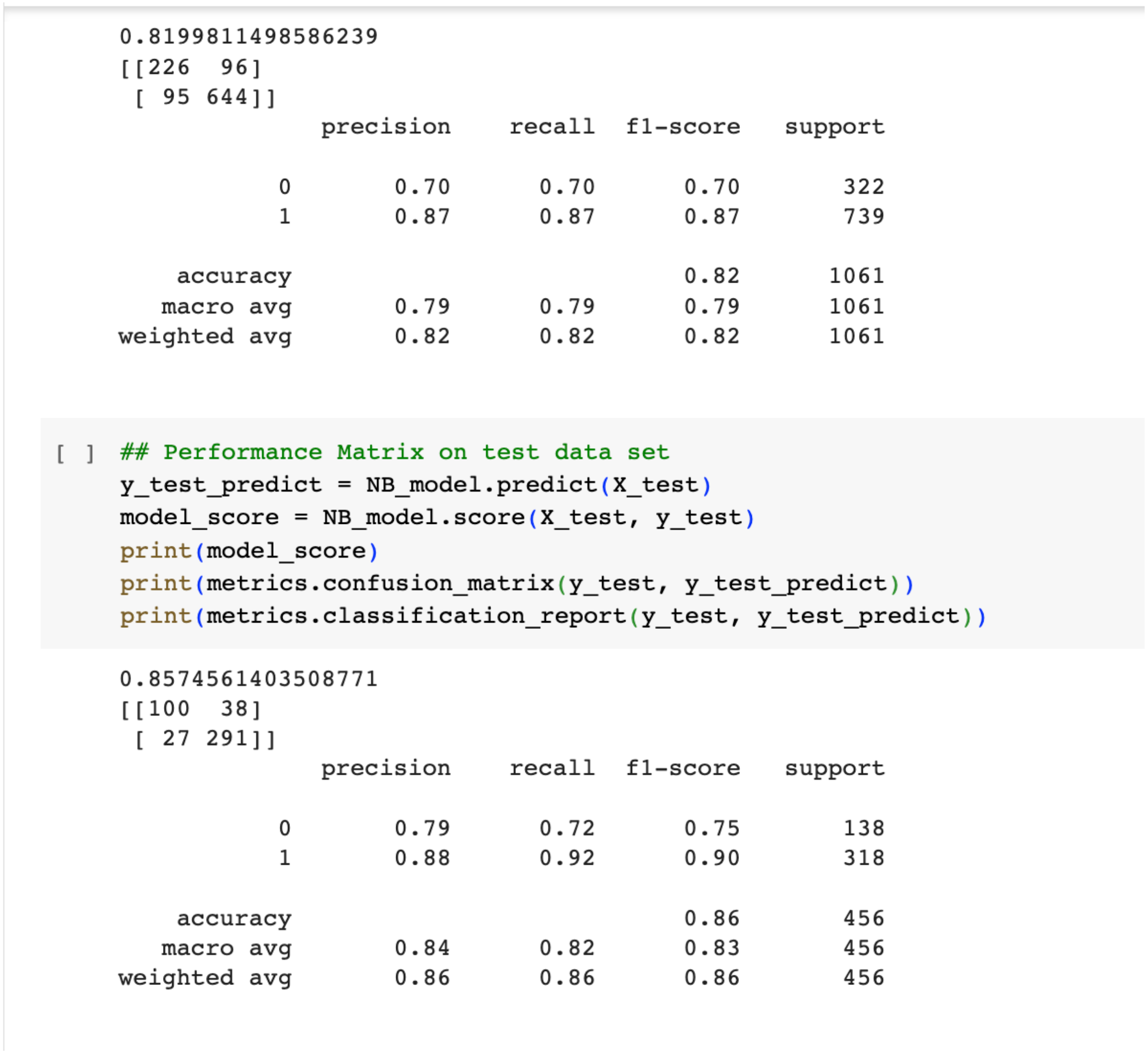
Naïve Bayes Model has also given similar range of accuracy on both training and test data.

Accuracy on train data: 82%

Accuracy on test data: 86%

AUC score for train: 0.87

AUC score for test:0.91



```
0.8199811498586239
[[226  96]
 [ 95 644]]
              precision    recall  f1-score   support

           0       0.70      0.70      0.70       322
           1       0.87      0.87      0.87       739

    accuracy                           0.82      1061
   macro avg       0.79      0.79      0.79      1061
weighted avg       0.82      0.82      0.82      1061
```

```
[ ] ## Performance Matrix on test data set
    y_test_predict = NB_model.predict(X_test)
    model_score = NB_model.score(X_test, y_test)
    print(model_score)
    print(metrics.confusion_matrix(y_test, y_test_predict))
    print(metrics.classification_report(y_test, y_test_predict))
```

```
0.8574561403508771
[[100  38]
 [ 27 291]]
              precision    recall  f1-score   support

           0       0.79      0.72      0.75       138
           1       0.88      0.92      0.90       318

    accuracy                           0.86       456
   macro avg       0.84      0.82      0.83       456
weighted avg       0.86      0.86      0.86       456
```

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

1.7 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model.

Bagging:

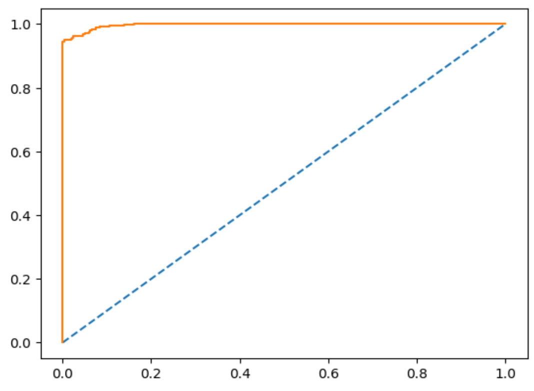
As bagging can reduce variance to avoid overfitting hence using RandomForestClssifier to check.

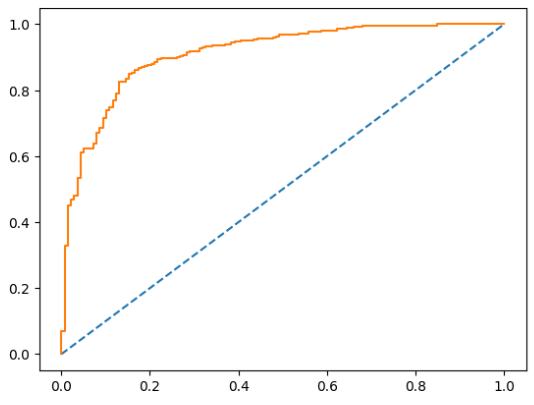
- As noticed that after bagging the model has predicted high on train set and just 85% accuracy on test, noticed the after bagging the prediction didn't change much.
- AUC score for train is hight 0.99 and test 0.91.

```
0.9679547596606974
[[298  24]
 [ 10 729]]
              precision    recall  f1-score   support

           0       0.97      0.93      0.95       322
           1       0.97      0.99      0.98       739

    accuracy                           0.97      1061
   macro avg       0.97      0.96      0.96      1061
weighted avg       0.97      0.97      0.97      1061
```

```
[ ] ## Performance Matrix on test data set
    y_test_predict = Bagging_model.predict(X_test)
    model_score = Bagging_model.score(X_test, y_test)
    print(model_score)
    print(metrics.confusion_matrix(y_test, y_test_predict))
    print(metrics.classification_report(y_test, y_test_predict))
```

```
0.8508771929824561
[[ 95  43]
 [ 25 293]]
              precision    recall  f1-score   support

           0       0.79      0.69      0.74       138
           1       0.87      0.92      0.90       318

    accuracy                           0.85       456
   macro avg       0.83      0.80      0.82       456
weighted avg       0.85      0.85      0.85       456
```





Boosting : will use AdaBoosting method here:

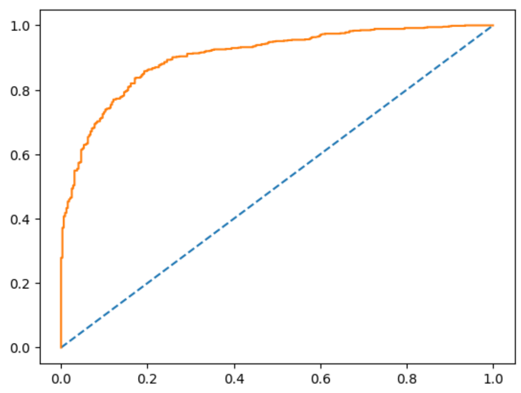
Accuracy for both Train and test remains as 85% and 84% respectably.

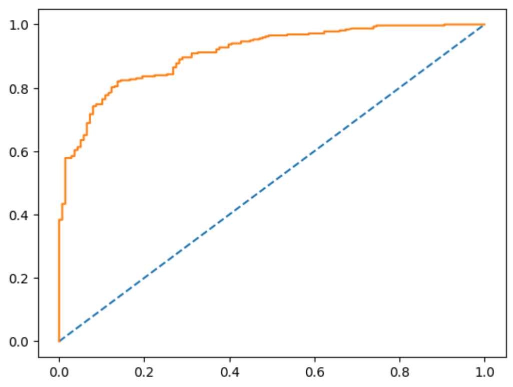
AUC score for both train and test is :090.

```
0.8491988689915174
[[228  94]
 [ 66 673]]
              precision    recall  f1-score   support

           0       0.78      0.71      0.74       322
           1       0.88      0.91      0.89       739

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

```
## Performance Matrix on test data set
y_test_predict = ADB_model.predict(X_test)
model_score = ADB_model.score(X_test, y_test)
print(model_score)
print(metrics.confusion_matrix(y_test, y_test_predict))
print(metrics.classification_report(y_test, y_test_predict))
```

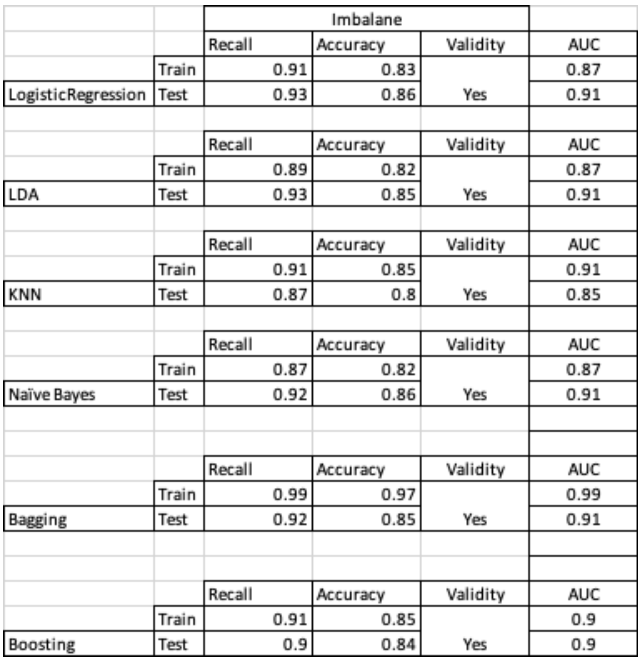
```
0.8355263157894737
[[ 95  43]
 [ 32 286]]
              precision    recall  f1-score   support

           0       0.75      0.69      0.72       138
           1       0.87      0.90      0.88       318

    accuracy                           0.84       456
   macro avg       0.81      0.79      0.80       456
weighted avg       0.83      0.84      0.83       456
```





Final Model: Compare the models and write inference which model is best/optimised.

All Model has performed well however logistic and LDA having little edge over others consider the AUC values.



		Imbalane			
		Recall	Accuracy	Validity	AUC
	Train	0.91	0.83		0.87
LogisticRegression	Test	0.93	0.86	Yes	0.91
		Recall	Accuracy	Validity	AUC
	Train	0.89	0.82		0.87
LDA	Test	0.93	0.85	Yes	0.91
		Recall	Accuracy	Validity	AUC
	Train	0.91	0.85		0.91
KNN	Test	0.87	0.8	Yes	0.85
		Recall	Accuracy	Validity	AUC
	Train	0.87	0.82		0.87
Naïve Bayes	Test	0.92	0.86	Yes	0.91
		Recall	Accuracy	Validity	AUC
	Train	0.99	0.97		0.99
Bagging	Test	0.92	0.85	Yes	0.91
		Recall	Accuracy	Validity	AUC
	Train	0.91	0.85		0.9
Boosting	Test	0.9	0.84	Yes	0.9

Inference:
1.8 Based on these predictions, what are the insights?

Based on the observations the LDA and Logical models are good to predict the exit polls.

As per the prediction on all models, majority of the population 35-60 having considerable political knowledge and are likely to vote for Labour party.

Project2-Machine Learning

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

President Franklin D. Roosevelt in 1941

President John F. Kennedy in 1961

President Richard Nixon in 1973

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

Total Characters:

Roosevelt= 7571

Kennedy = 7618

Nixon= 9991

Total Words:

Roosevelt= 1536

Kennedy = 1546

Nixon= 2028

Total Sentences:

Roosevelt= 68

Kennedy = 52

Nixon= 69

2.2 Remove all the stopwords from all three speeches.

```
Words1= list(inaugural.words('1941-Roosevelt.txt'))
Words2= list(inaugural.words('1961-Kennedy.txt'))
Words3= list(inaugural.words('1961-Kennedy.txt'))
```

1. Cratered a separate list for all 3 files.
2. Created a function with all stop words
3. Applied the same in each files.

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

President Franklin D. Roosevelt in 1941

1. **Sprit**
2. **2. Life**
3. **Democracy**

President John F. Kennedy in 1961

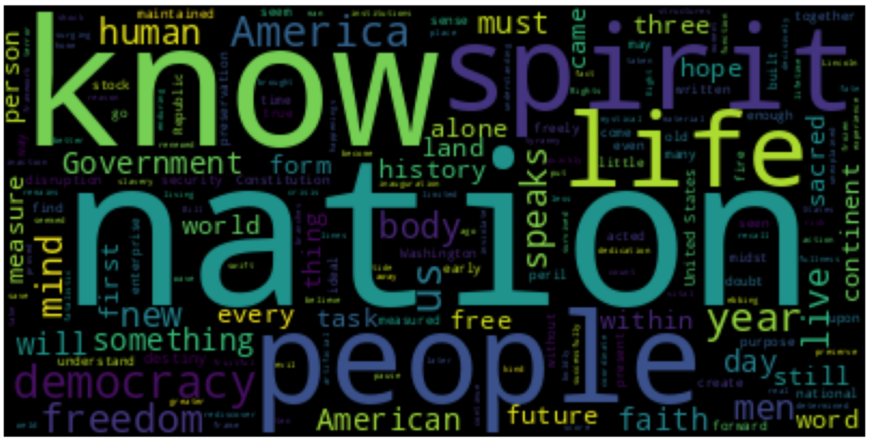
1. **World**
2. **Pledge**
3. **Citizens**

President Richard Nixon in 1973

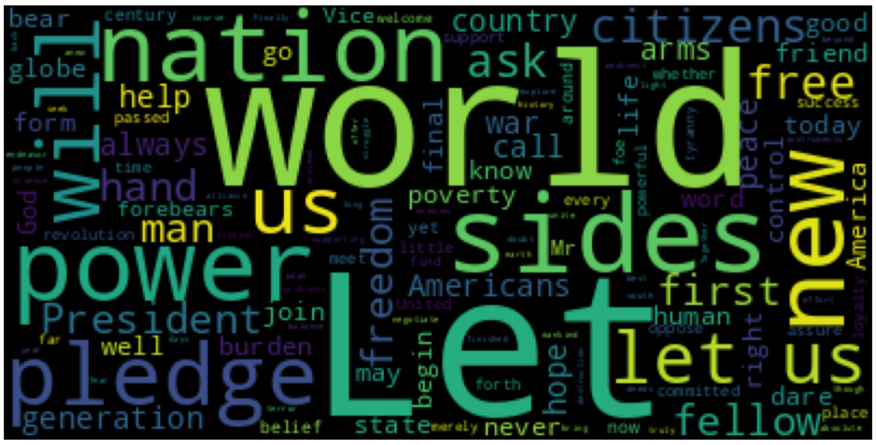
1. **World**
2. **Citizens**
3. **Power**

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

President Franklin D. Roosevelt



President John F. Kennedy in 1961



President Richard Nixon in 1973

world
Let
nation
sides
power
pledge
new
will
us
let us
citizens
country
ask
free
arms
friend
good
today
help
always
hand
man
forebears
poverty
know
call
war
final
freedom
Americans
President
first
human
generation
fellow
America
peace
right
hope
begin
may
burden
well
join
state
never
belief
dare
bear
century
Vice
globe
form
God