BUSINESS DATA

ANALYSIS REPORT

Table of Contents

Problem1:

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

Dataset for Problem: Election_Data.xlsx

Data 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.

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 data is imported and the following are the observations:

	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

	Unnamed: 0	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
1520	1521	Conservative	67	5	3	2	4	11	3	male
1521	1522	Conservative	73	2	2	4	4	8	2	male
1522	1523	Labour	37	3	3	5	4	2	2	male
1523	1524	Conservative	61	3	3	1	4	11	2	male
1524	1525	Conservative	74	2	3	2	4	11	0	female

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

- The data-set has 1525 rows and 10 columns. There are 2 object type data types and rest 8 variables are int data types
- 'age' variable is a continuous variable whereas other integer variables are ordinal variables
- 'vote' and 'gender' are categorical variables
- 'vote' is the dependent variable that needs to be predicted
- There are no missing values in variables
- Column Unnamed: 0 is a redundant variable and can be dropped .
- After dropping the column Unnamed: 0 we see that there are **8** duplicates in the data.

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

- We can drop the duplicate data as part of data clean-up.

```
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   vote                    1517 non-null   object
 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   object
dtypes: int64(7), object(2)
```

- After dropping the duplicate rows there are 1517 observations and 9 columns and There are 2 object type data types and 7 variables are int data types

- Let's see data summary of the categorical and nominal variables

	count	unique	top	freq
vote	1517	2	Labour	1057
economic.cond.national	1517	5	3	604
economic.cond.household	1517	5	3	645
Blair	1517	5	4	833
Hague	1517	5	2	617
Europe	1517	11	11	338
political.knowledge	1517	4	2	776
gender	1517	2	female	808

  - Based on this data-set we see that Labour party is getting maximum votes with 1057 votes out of 1517 voters

- Assessment of current national economic conditions has a mean of 3 with 604 out of 1517 voters' assessment of current national economic conditions is 3 on scale of 1 to 5
- Assessment of current household economic conditions has a mean of 3 with 645 out of 1517 voters' assessment of current household economic conditions is 3 on scale of 1 to 5
- Assessment of the Labour leader, Blair has an average rating of 4, with 833 of 1517 voters when asked for assessment of the Labour leader, Blair, rated him 4 on a scale of 1 to 5.
- Assessment of the Conservative leader, Hague has an average rating of 2 , with 617(majority of voters) of 1517 voters when asked for assessment of the Conservative leader, Hague, rated him 2 on a scale of 1 to 5
- On an 11-point scale that measures respondents' attitudes toward European integration has an average rating of almost 7. 338 of 1517 voters scored a 11 meaning they are highly Eurosceptic .
- Voters political knowledge of parties' positions on European integration is on an average 1.5 Majority of the voters (776 of them out of 1517) when assessed on Knowledge of parties' positions on European integration showed a political knowledge of 2 on a scale of 0 to 3.
  0 being No knowledge and 3 being highly aware of Knowledge of parties' positions on European integration.
- Based on this data-set we see that there are more female than male voters with 808 votes out of 1517 voters being Female

- Let's see data summary of the numerical variable age

	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

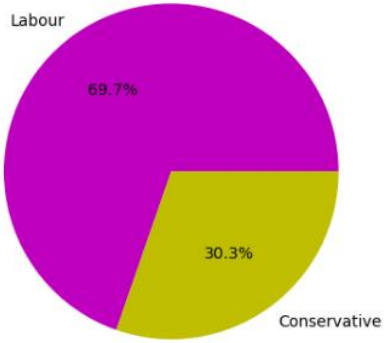
- From the descriptive summary of variable 'age' ,it can be inferred that the average age of the voters is 54 years and median is 53 years, which means that variable 'age' is normally distribution .The youngest of the voter is of 24 years old and the eldest voter is 93 years old • 50% of the voters are between the age of 41 and 67 years.

•Let's see the skewness of the numerical variables

```
age                   0.139800
economic.cond.national   -0.238474
economic.cond.household   -0.144148
Blair                 -0.539514
Hague                  0.146191
Europe                -0.141891
political.knowledge      -0.422928
```

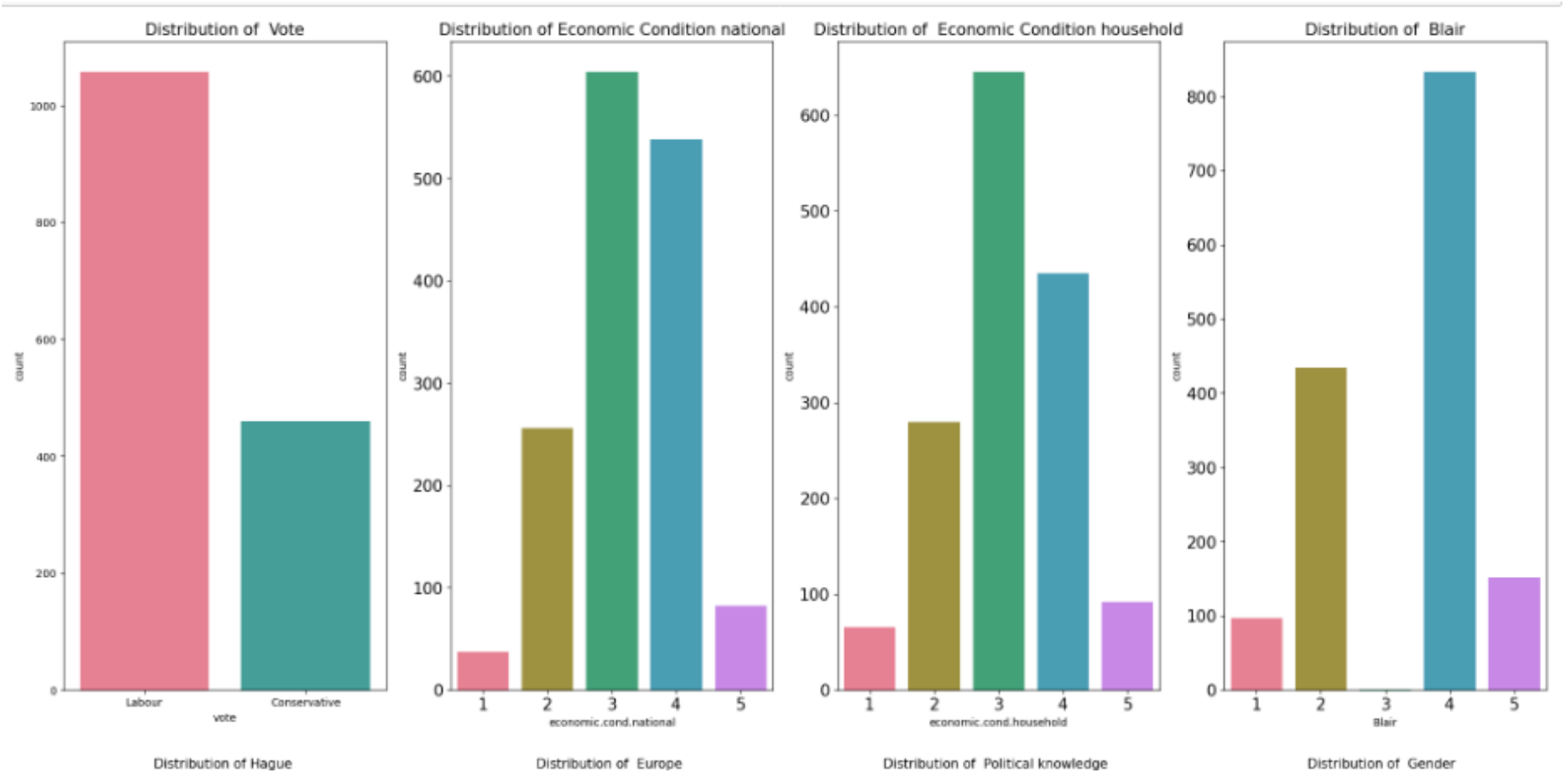
As we can see from skewness of the variables from the table above, the age column and the Hague column are positively skewed and all the other columns are negatively skewed. 'Blair' is the most negatively skewed column followed by the 'political.knowledge' column and 'economic.cond.national' column .

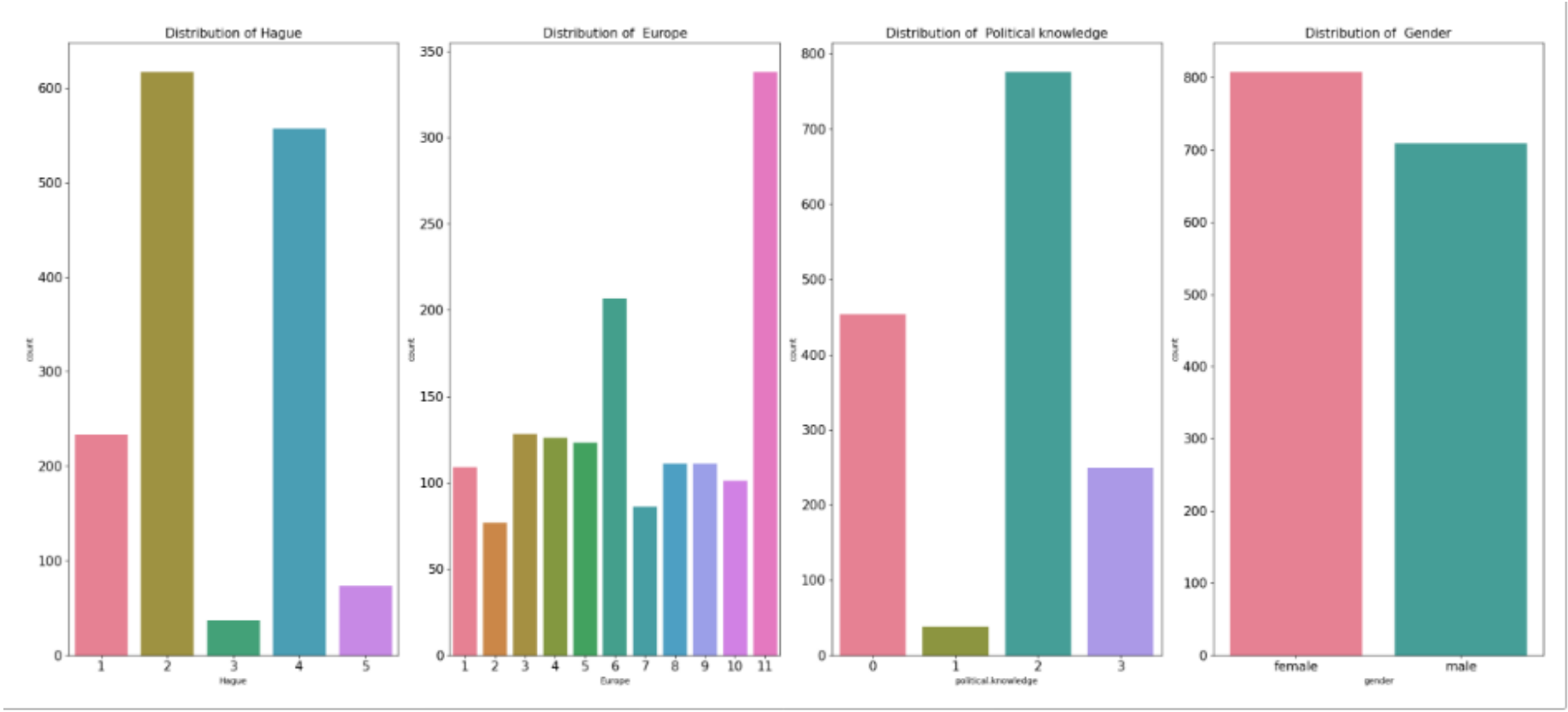
1.2 Perform EDA (Check the null values, Data types, shape, Univariate, bivariate analysis). Also check for outliers (4 pts). Interpret the inferences for each (3 pts) Distribution plots(histogram) or similar plots for the continuous columns. Box plots. Appropriate plots for categorical variables. Inferences 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.



There are 69. 7% voters for Labour party and 30.3 % voters for Labour party

There are no null values in the dataset . Followingis the value counts of each category






```
Number of Unique Values of  VOTE : 2


Value Counts of  vote
Conservative     460
Labour          1057
Name: vote, dtype: int64


Percentage values of each category of  vote
```

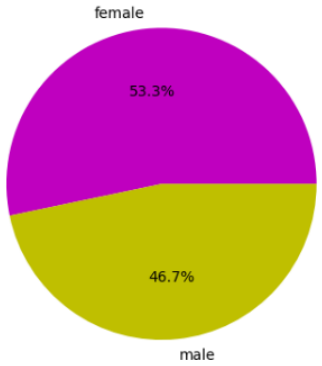
```
Conservative    0.30323
Labour          0.69677
Name: vote, dtype: float64


------------------------------------------------------------
Number of Unique Values of  ECONOMIC.COND.NATIONAL : 5


Value Counts of  economic.cond.national
1     37
5     82
2    256
4    538
3    604
Name: economic.cond.national, dtype: int64


Percentage values of each category of  economic.cond.national
1    0.024390
5    0.054054
2    0.168754
4    0.354647
3    0.398154
Name: economic.cond.national, dtype: float64


------------------------------------------------------------
Number of Unique Values of  ECONOMIC.COND.HOUSEHOLD : 5


Value Counts of  economic.cond.household
1     65
5     92
2    280
4    435
3    645
```

```
Name: economic.cond.household, dtype: int64


Percentage values of each category of  economic.cond.household
1    0.042848
5    0.060646
2    0.184575
4    0.286750
3    0.425181
Name: economic.cond.household, dtype: float64


------------------------------------------------------------
Number of Unique Values of  BLAIR : 5


Value Counts of  Blair
3      1
1     97
5    152
2    434
4    833
Name: Blair, dtype: int64


Percentage values of each category of  Blair
3    0.000659
1    0.063942
5    0.100198
2    0.286091
4    0.549110
Name: Blair, dtype: float64


------------------------------------------------------------
Number of Unique Values of  HAGUE : 5
```

```
Value Counts of  Hague
3     37
5     73
1    233
4    557
2    617
Name: Hague, dtype: int64


Percentage values of each category of  Hague
3    0.024390
5    0.048121
1    0.153593
4    0.367172
2    0.406724
Name: Hague, dtype: float64


------------------------------------------------------------
Number of Unique Values of  EUROPE : 11


Value Counts of  Europe
2      77
7      86
10    101
1     109
9     111
8     111
5     123
4     126
3     128
6     207
11    338
Name: Europe, dtype: int64
```

```
Percentage values of each category of  Europe
2     0.050758
7     0.056691
10    0.066579
1     0.071852
9     0.073171
8     0.073171
5     0.081081
4     0.083059
3     0.084377
6     0.136454
11    0.222808
Name: Europe, dtype: float64


----------------------------------------------------------
Number of Unique Values of  POLITICAL.KNOWLEDGE : 4


Value Counts of  political.knowledge
1     38
3    249
0    454
2    776
Name: political.knowledge, dtype: int64


Percentage values of each category of  political.knowledge
1    0.025049
3    0.164140
0    0.299275
2    0.511536
Name: political.knowledge, dtype: float64


----------------------------------------------------------
```

```
Number of Unique Values of  GENDER : 2


Value Counts of  gender
male      709
female    808
Name: gender, dtype: int64


Percentage values of each category of  gender
male      0.46737
female    0.53263
Name: gender, dtype: float64


------------------------------------------------------------
Number of Unique Values of  AGE_CAT : 5


Value Counts of  age_cat
5     62
1    199
4    366
3    415
2    475
Name: age_cat, dtype: int64


Percentage values of each category of  age_cat
5    0.040870
1    0.131180
4    0.241266
3    0.273566
2    0.313118
Name: age_cat, dtype: float64
```



There re 53.3% female voters and 46.7% male voters

• The distribution of variable 'age' in years is slightly positively skewed and is multimodal .The boxplot of variable 'age' is slightly positively skewed and there are no outliers and 75% of the voters age is below 67.

• The distribution of Assessment of current national economic conditions shown by variable economic.cond.national ranging 1 to 5 is negatively skewed and is multimodal. The box plot of economic.cond. national has outliers and 1 and 5 are very few in number compared to 2,3 and 4

The distribution of Assessment of current household economic conditions shown by variable economic.cond.household ranging 1 to 5 is negatively skewed and is multimodal. The box plot of economic.cond.household has outliers and 1 and 5 are very few in number compared to 2,3 and 4

• The distribution of Assessment of the Labour leader shown by variable 'Blair' has range 1 to 5 is highly negatively skewed and is multimodal. The box plot of 'Blair' has no outliers and 4 occurs very frequently and 3 are very few in number

• The distribution of Assessment of the Conservative leader shown by variable 'Hague' has range 1 to 5 is positively skewed and is multimodal. The box plot of 'Hague' has no outliers and 2,4 occurs very frequently and 3 are very few in number

• The distribution of an 11-point scale that measures respondents' attitudes toward European integration represented by variable 'Europe' is negatively skewed. Many respondents seem to have scaled European integration at 11, Since High scores represent 'Eurosceptic' sentiment. Majority of the respondents seems to be Eurosceptic'

• The distribution of Knowledge of parties' positions on European integration, 0 to 3 represented by variable 'political.knowledge' is negatively skewed. The box plot of 'political.knowledge' has no outliers and 2 occurs very frequently and there are no records for 0

Bivariate Analysis-

Strip plot of Categorical Variable Vote Vs Numerical Variables

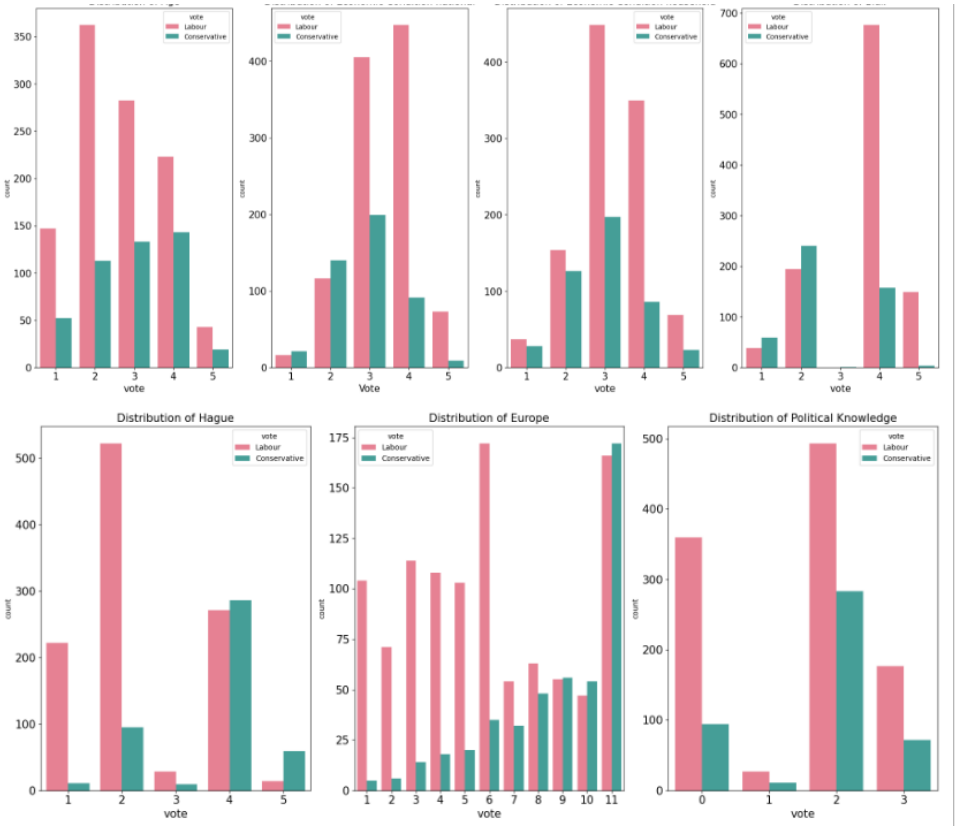


Figure 1

We can infer from the above Countplots:

- The of 'age' Vs 'vote' indicates that there are more Labour than Conservative party voters in a given age group in the dataset . The minimum age for both are 24
- The of 'economic.cond.national Vs 'vote' indicates that there fewer voters supporting Labour vote with type 1 and fewer voters supporting Conservative vote with type 5
- The of 'economic.cond.household Vs 'vote' indicates that there fewer voters supporting Labour vote with type 1 and fewer voters supporting Conservative vote with type 5

- The plot of 'Blair' Vs 'vote' indicates that there no voters supporting Labour vote with type 3 and fewer voters supporting Conservative vote with type 3 & 5
- The plot of 'Hague' Vs 'vote' indicates that there no voters supporting Labour vote with type 3 & 5 and fewer voters supporting Conservative vote with type 1 & 3
- The plot of 'Europe' Vs 'vote' indicates that there is more percentage of voters in conservative group than Labour group who are Eurosceptic
- The plot of 'political.knowledge' Vs 'vote' indicates that there are fewer voters supporting Labour vote with type 1 and fewer voters supporting Conservative vote with type 1

Plot of Categorical Variable Gender Vs Variables

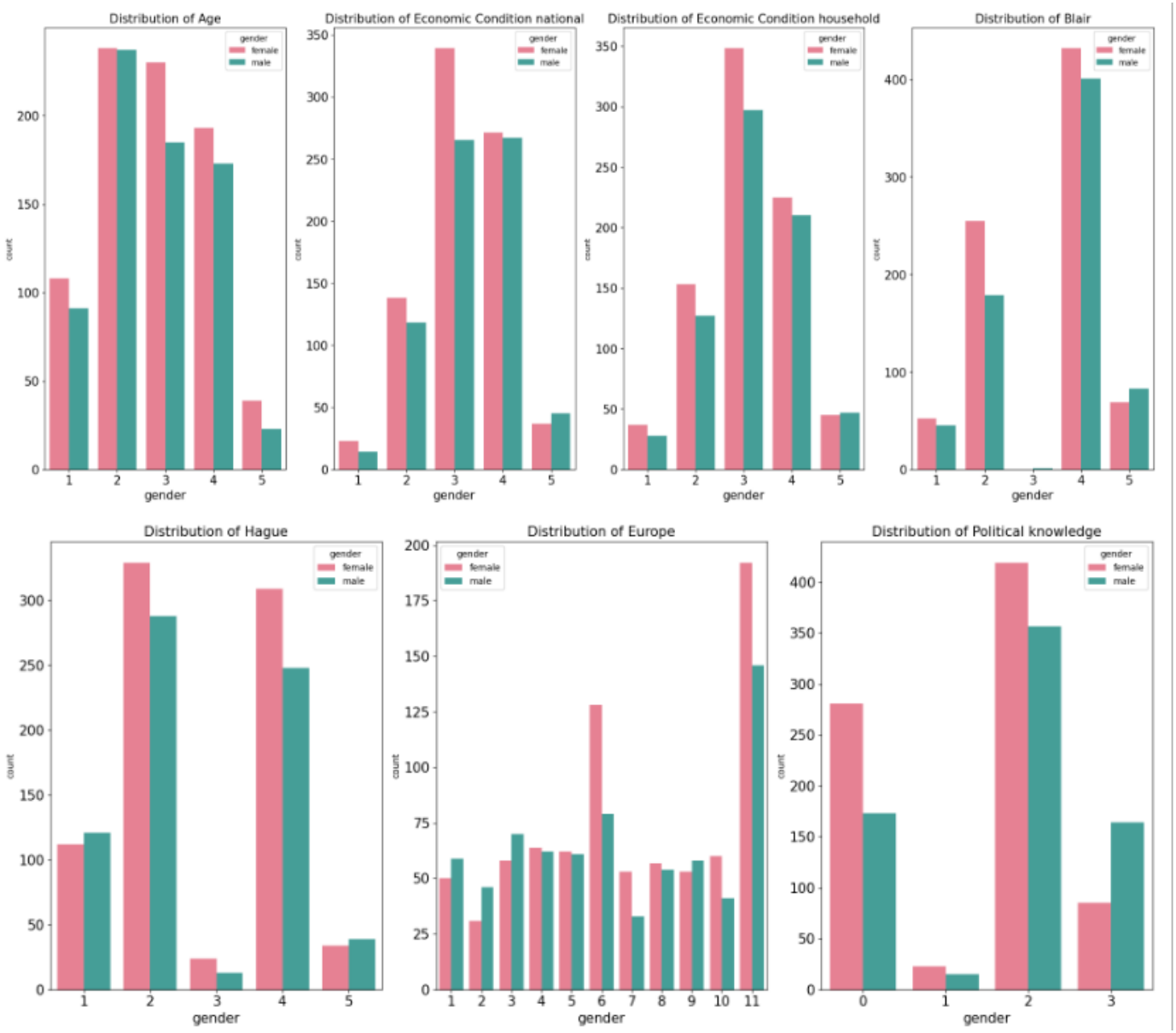

Distribution of Age
Distribution of Economic Condition national
Distribution of Economic Condition household
Distribution of Blair
Distribution of Hague
Distribution of Europe
Distribution of Political knowledge
gender
female
male
count
gender

Figure 2

We can infer from the above:

- The plot of 'economic.cond.national Vs 'gender' indicates that there fewer voters supporting Labour vote with type 1 and fewer voters supporting Conservative vote with type 5
- The plot of 'economic.cond.household Vs 'gender' indicates that there fewer voters supporting Labour vote with type 1 and fewer voters supporting Conservative vote with type 5
- The plot of 'Blair' Vs 'gender' indicates that there no voters supporting Labour vote with type 3 and fewer voters supporting Conservative vote with type 3 & 5
- The plot of 'Hague' Vs 'gender' indicates that there no voters supporting Labour vote with type 3 & 5 and fewer voters supporting Conservative vote with type 1 & 3
- The plot of 'Europe' Vs 'gender' indicates that there is more percentage of voters in conservative group than Labour group who are Eurosceptic
- The plot of 'political.knowledge' Vs 'gender' indicates that there are fewer voters supporting Labour vote with type 1 and fewer voters supporting Conservative vote with type 1.

Correlation heatmap of the numerical variables :

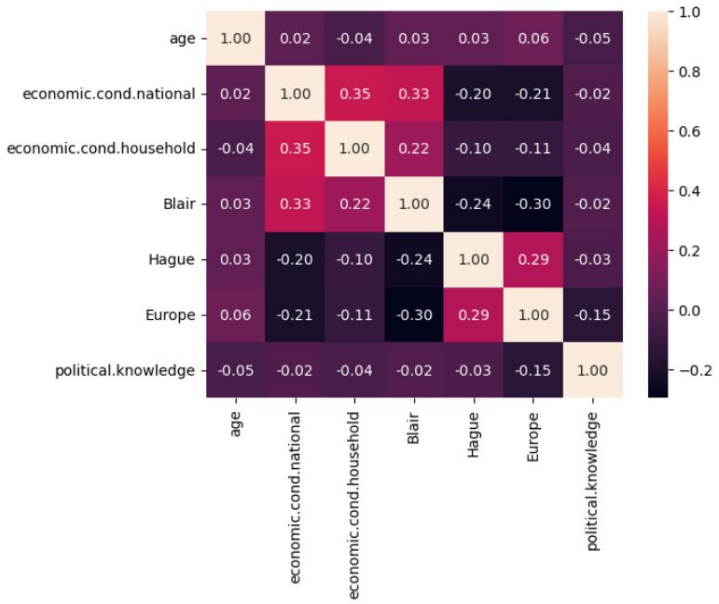


Figure 3

• From the above correlation plot, we can see that the variables economic.cond.household , economic.cond.national ,Blair are weakly correlated to each other. But we can't infer anything from the heatmap because all the numerical variables except age are categorical variables having

Pairplot of the numerical variables with hue as Vote

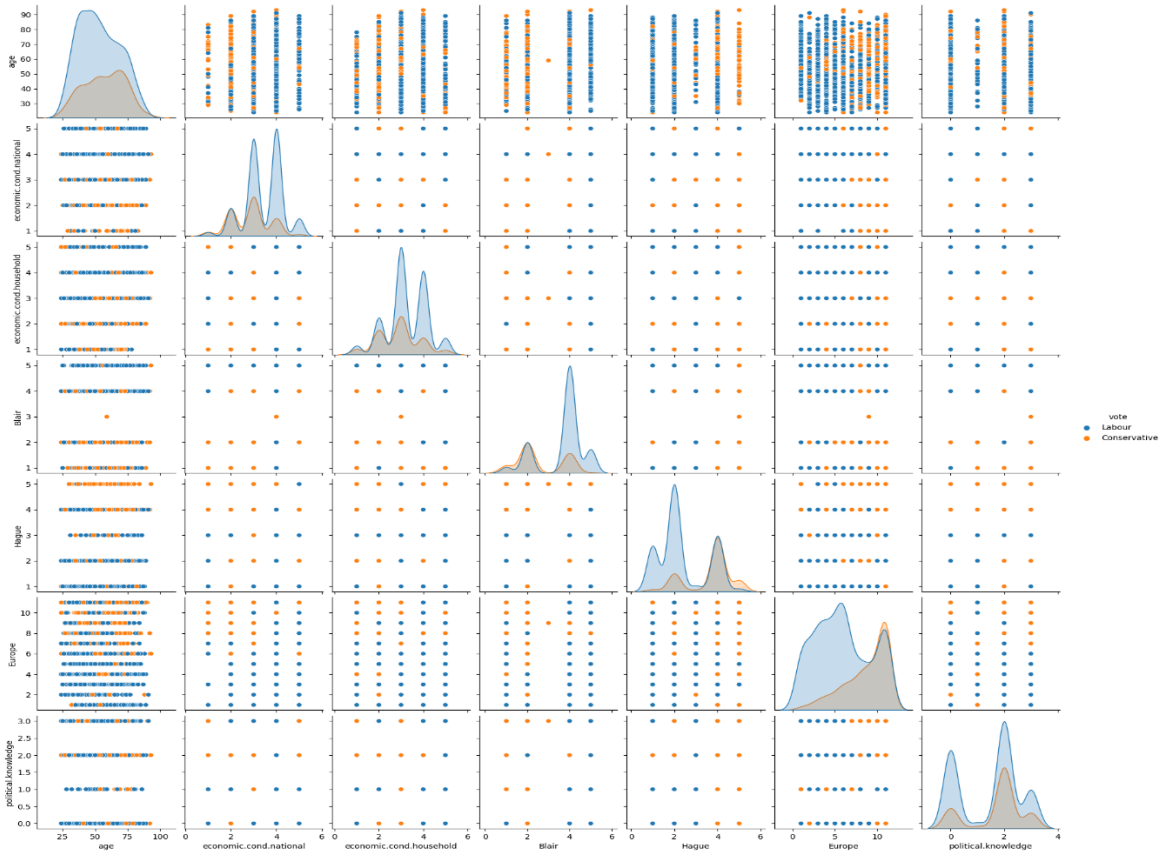


Figure 4

From the pairplot shown above, we can see that there is no correlation between the variables:

• The scatter plots does not say a lot about the variables, but from the distribution we can see that 'Europe' is having 2 peaks with Labour Vote and one peak with Conservative Vote

EDA

• The age distribution of voters follows a normal distribution with mean of 54 years and median of 53 years. There are no outliers

- A voter between 30 and 50 years is more likely to vote for Labour, whereas a voter between 65 and 75 years is more likely to be a conservative voter
- A voter below 30 is not as decisive as other age groups as they are more unlikely to vote in the election than those above
- Those rated national and household economic condition below 2 are an exception, thus outliers
- A voter who rated the economic conditions to be lower is more likely to vote for Conservative, where as those rated high may vote for Labour
- Mr Blair got higher acceptance among both Labour and Conservative voters, whereas Mr Hague got poorly rated among Labour
- Mr Hague got higher acceptance from the Conservatives
- A highly Eurosceptic voter is highly probable to vote for Conservatives, whereas a lesser Eurosceptic voter is more likely to vote for Labour
- For a Labour voter the flourishing economic condition and acceptance of Mr Blair as Prime Minister is observed to be major a motivation to vote for Labour
- Whereas concerns on EU's increasing influence appears to be the major consideration for a Conservative vote

Outliers check

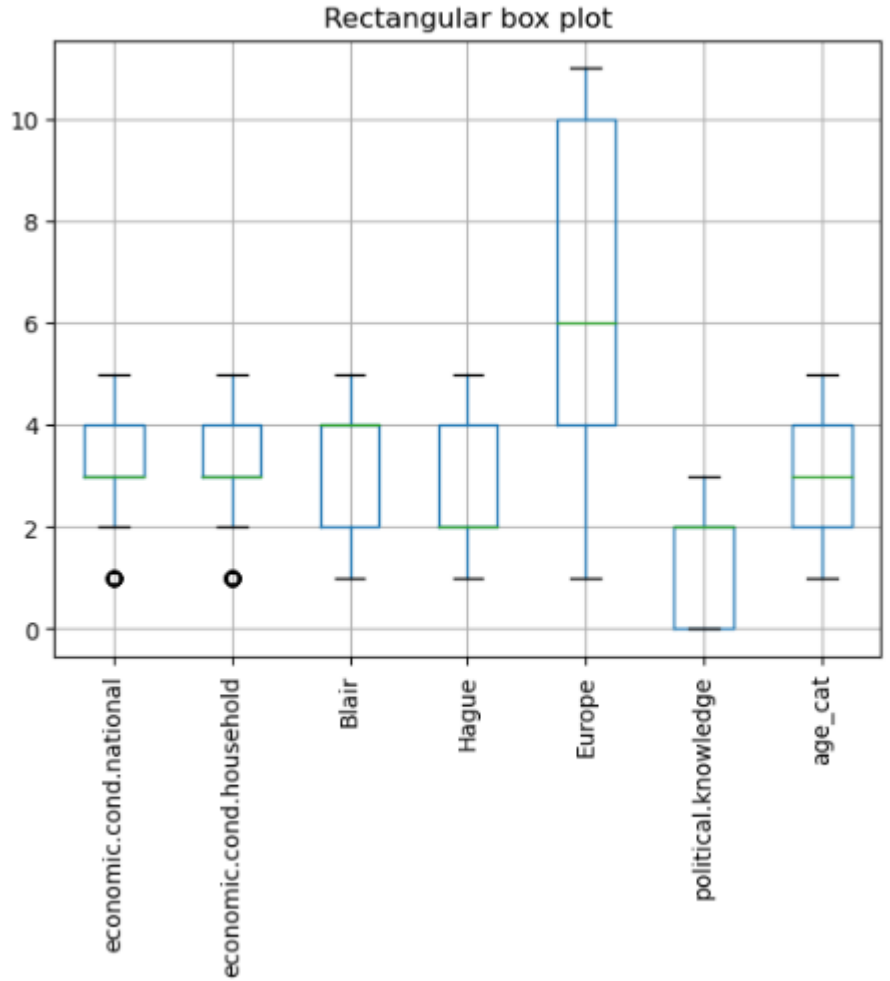


Figure 5

Outlier Detection & Treatment

• The black circles in boxplots show that there is presence of outliers the variables economic.cond.household and economic.cond.national . Gaussian Naive Bayes model is highly impacted by outliers so it makes sense to remove outliers.

• All the outliers are treated by adjusting them to the lower and upper bound values calculated by the IQR value(Q3-Q1.

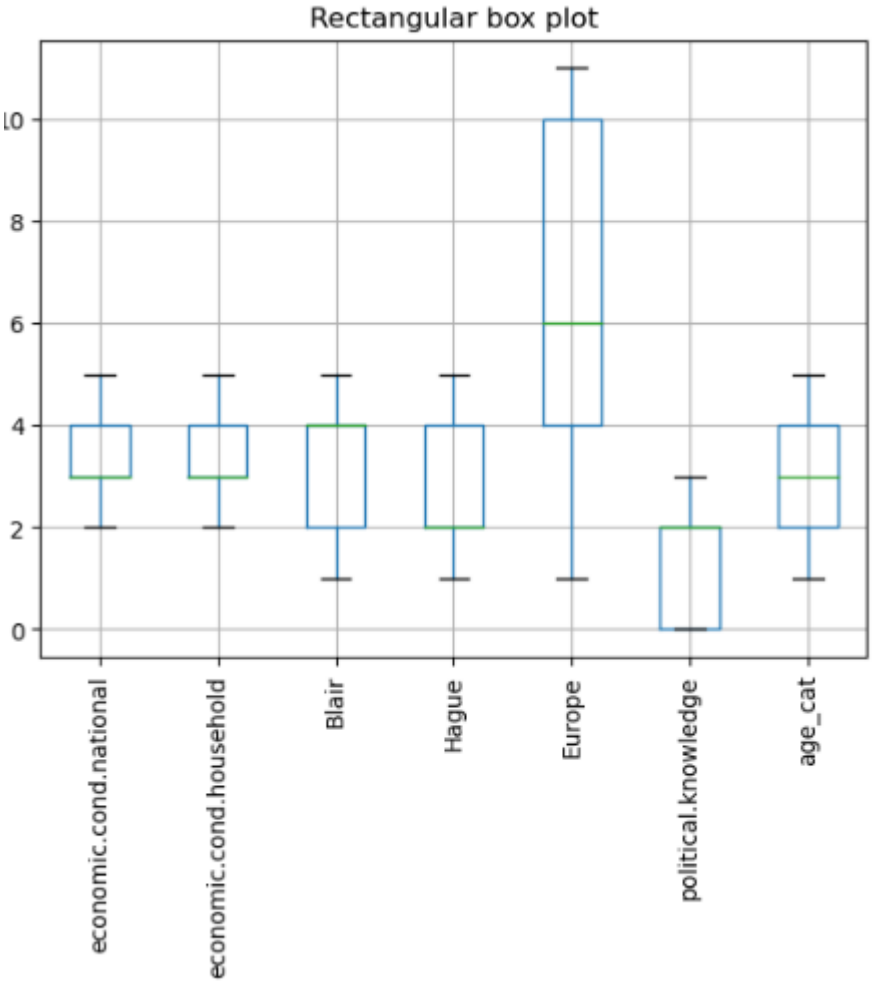


Figure 6

All the outliers are now treated and as we can see now there are no outliers

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

Encoding

There are 2 categorical variables 'vote' and 'gender' , we can encode the categorical variables using One Hot encoding. A new column is created, with 1 indicating that variable as True and 0 as False and this is how the extended variable's data looks. If the vote_Labour=1 indicates it's a vote for Labour and 0 indicates vote for Conservative. If the gender_male=1 indicates it's a male voter and 0 indicates female voter. For ease of model building age variable is bucket into 5 buckets [0,-35,35-50,50-65,65-80,80-100] with labels '1', '2', '3','4','5' respectively Age variable is dropped and instead age_cat variable that was created as categorical variable is added

	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	age_cat	vote_Labour	gender_male
0	3	3	4	1	2	2	2	1	0
1	4	4	4	4	5	2	2	1	1
2	4	4	5	2	3	2	1	1	1
3	4	2	2	1	4	0	1	1	0
4	2	2	1	1	6	2	2	1	1

Scaling

As we can see from the density plot above all variables are cantered around 0.

Splitting the Dataset

We need to split a dataset into train and test sets to evaluate how well our machine learning model performs. We split the dataset into training and testing dataset in a ratio of 70:30.The train set is used to fit the model, and the statistics of the train set are known. The second set is called the test data set, this set is solely used for predictions.

"vote"' is the column we've to predict so we take that column as y and the rest of the columns as our X variable. 30% of the whole data is taken as our test set, and 70% as our train set. The random state is set to 1 that helps us get the same random split each time.

Number of rows in training dataset is 1061 and number of columns in training dataset is 8.
Number of rows in testing dataset is 456 and number of columns in testing dataset is 8.
Number of rows in target variable in training dataset is 1061 and 1 column 'vote'.
Number of rows in target variable in testing dataset is 456 and 1 column 'vote'
Top 5 records of Training Dataset

	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	age_cat	gender_male
987	2.0	4.0	1	4	11	2	1	0
1267	4.0	3.0	4	4	6	0	2	1
647	4.0	3.0	4	4	7	2	3	0
675	3.0	3.0	4	2	11	0	2	1
537	5.0	3.0	4	2	8	0	2	1

Figure 7

	vote_Labour
987	0
1267	1
647	0
675	1
537	1

Figure 8

Top 5 records of Testing Dataset

	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	age_cat	gender_male
503	3.0	3.0	2	2	8	2	4	0
368	3.0	2.0	4	2	8	3	2	1
1071	5.0	5.0	5	2	1	2	5	1
1027	2.0	3.0	2	4	8	2	2	0
1322	5.0	4.0	4	4	8	0	1	1

As we can see that dataset is randomly split between the training and testing datasets

1.4) Apply Logistic Regression and LDA (Linear Discriminant Analysis) (2 pts). Interpret the inferences of both model s (2 pts). Successful implementation of each model. Logical reason should be shared if any custom changes are made to the parameters while building the model. Calculate Train and Test Accuracies for each model. Comment on the validness of models (over fitting or under fitting)

Linear Discriminant Analysis

Linear Discriminant Analysis with default parameters :

Let's build an Linear Discriminant Analysis model with default hyper parameters

Parameters	Value
solver	svd
tol	0.0001

solver - 'svd': Singular value decomposition (default). Does not compute the covariance matrix, therefore this solver is recommended for data with a large number of features.

Tolerance - 0.0001: Absolute threshold for a singular value of X to be considered significant, used to estimate the rank of X. Dimensions whose singular values are non-significant are discarded. Only used if solver is 'svd'.

	precision	recall	f1-score	support
0	0.74	0.66	0.70	307
1	0.87	0.91	0.89	754
accuracy			0.84	1061
macro avg	0.81	0.78	0.79	1061
weighted avg	0.83	0.84	0.83	1061

	precision	recall	f1-score	support
0	0.77	0.75	0.76	153
1	0.87	0.89	0.88	303
accuracy			0.84	456
macro avg	0.82	0.82	0.82	456
weighted avg	0.84	0.84	0.84	456

The above table shows the Accuracy, Precision, Recall and the F1 of the LDA model for both the train and the test data.

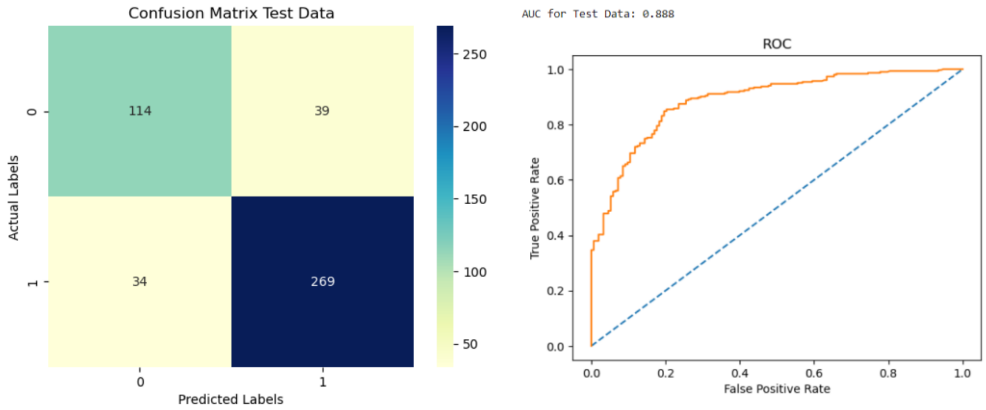
• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 87% for both the train and the test data for classifier 1 and 74% for training and 76% for testing data for classifier 0.

• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 91% for the train data and 89% for the test data for classifier 1 and 66% for training and 75% for testing data for classifier 0.

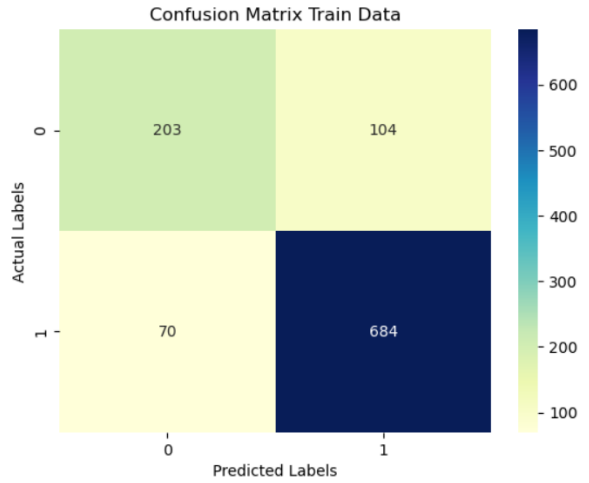
• **F1-score** : It combines the precision and recall of a classifier into a single metric by taking their harmonic mean. It is primarily used to compare the performance of two classifiers. Suppose that classifier A has a higher recall, and classifier B has higher precision. In this case, the F1-scores for both the classifiers can be used to determine which one produces better results. The greater the F1 Score, the better is the performance of our model. Here it is 89% for train data and 88% for test data for classifier 1 and 70% for train data and 76% for test data for classifier 0

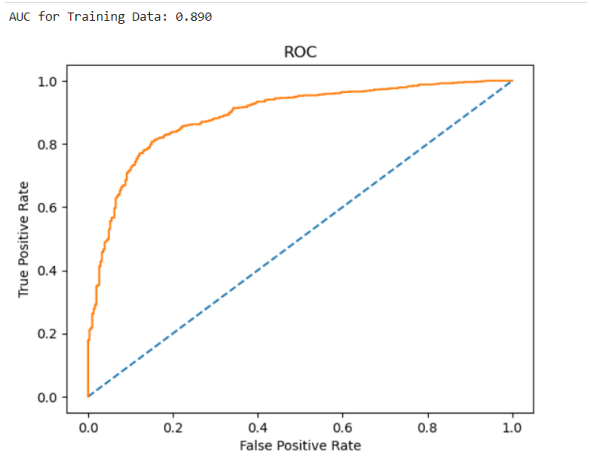
• **Accuracy**: The train data accuracy is 84% and the test data accuracy is 84%.

When a model performs very well in the train data and but doesn't perform well in the testing set performs ,it's called overfitting and the inverse is called the underfitting. In this case the model accuracy is same for both training and testing dataset so there is no over or underfitting . It's a good Model



We can from the confusion matrix above see that 114 of 153 of class 0 are precited correctly and 269 of 303 of class 1 are predicted correctly in testing data





We can from the confusion matrix above see that 203 of 307 of class 0 are precited correctly and 684 of 754 of class 1 are predicted correctly

AUC scores for both testing is 0.888 and training data is 89 is almost same , so there is no overfitting

Logistic Regression

Let's build Logistic Regresion model with following hyper parameters

Parameters	Value
solver	newton-cg
tol	0.0001
max_iter	10000
n_jobs	2
penalty	'none'
random_state	1

solver . Algorithm to use in the optimization problem. Default is 'lbfgs'.To choose a solver, you might want to consider the following aspects:

- For small datasets, 'liblinear' is a good choice, whereas 'sag' and 'saga' are faster for large ones;

- For multiclass problems, only 'newton-cg', 'sag', 'saga' and 'lbfgs' handle multinomial loss;

- 'liblinear' is limited to one-versus-rest schemes.

max_iter : int, default=100 Maximum number of iterations taken for the solvers to converge.

n_jobs : int, default=None .Number of CPU cores used when parallelizing over classes if multi_class='ovr'". This parameter is ignored when the ``solver`` is set to 'liblinear' regardless of whether 'multi_class' is specified or not. ``None`` means 1 unless in a :obj:`joblib.parallel_backend` context. ``-1`` means using all processor s.

penalty : {'l1', 'l2', 'elasticnet', 'none'}, default='l2' Specify the norm of the penalty. `'none'`: no penalty is added

```
              precision    recall  f1-score   support

           0       0.75      0.64      0.69       307
           1       0.86      0.91      0.89       754

    accuracy                           0.84      1061
   macro avg       0.81      0.78      0.79      1061
weighted avg       0.83      0.84      0.83      1061
```

```
              precision    recall  f1-score   support

           0       0.76      0.73      0.74       153
           1       0.87      0.88      0.87       303

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

The above table shows the Accuracy, Precision, Recall and the F1 of the Logistic Regression model for both the train and the test data.

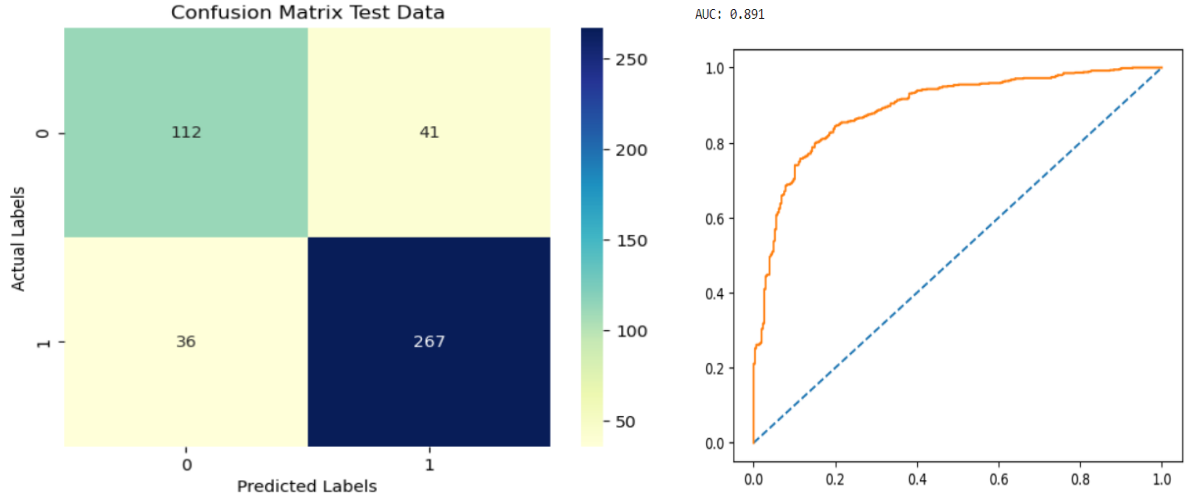
• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 86% for train and 87 % the test data for classifier 1 and 75% for training and 76% for testing data for classifier 0.

• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 91% for the train data and 88% for the test data for classifier 1 and 64% for training and 73% for testing data for classifier 0.

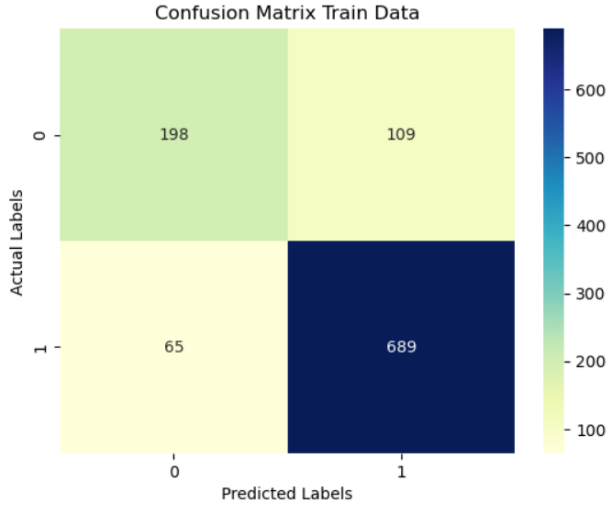
• **F1-score** : It combines the precision and recall of a classifier into a single metric by taking their harmonic mean. It is primarily used to compare the performance of two classifiers. Suppose that classifier A has a higher recall, and classifier B has higher precision. In this case, the F1-scores for both the classifiers can be used to determine which one produces better results. The greater the F1 Score, the better is the performance of our model. Here it is 89% for train data and 87% for test data for classifier 1 and 69% for train data and 74% for test data for classifier 0

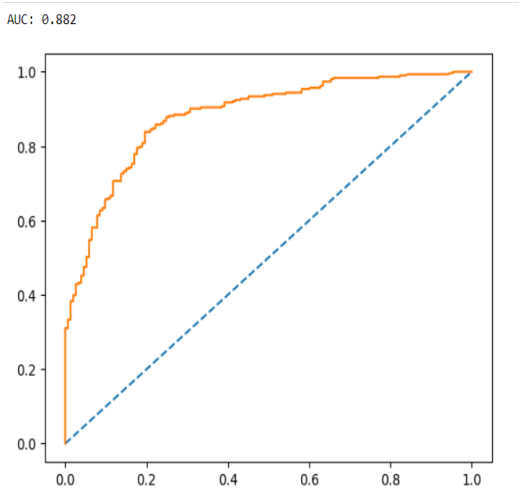
• **Accuracy**: The train data accuracy is 84% and the test data accuracy is 83%.

When a model performs very well in the train data and but doesn't perform well in the testing set performs ,it's called overfitting and the inverse is called the underfitting. In this case the model accuracy is same for both training and testing dataset so there is no over or underfitting . It's a good Model



We can from the confusion matrix above see that 112 of 153 of class 0 are precited correctly and 267 of 303 of class 1 are predicted correctly in testing data





We can from the confusion matrix above see that 198 of 307 of class 0 are precited correctly and 689 of 754 of class 1 are predicted correctly

AUC scores for testing is 0.882 and training data is 0.891 and is very close , so there is no overfitting

1.5) Apply KNN Model and Naïve Bayes Model (2pts). Interpret the inferences of each model (2 pts). Successful implementation of each model. Logical reason should be shared if any custom changes are made to the parameters while building the model. Calculate Train and Test Accuracies for each model. Comment on the validness of models (over fitting or under fitting)

KNN:

Let's build KNN model with default hyper parameters

Parameters	Value
n_neighbors	5
weights	uniform
Metric	'minkowski'
n_jobs	None

n_neighborsint, default=5.Number of neighbours to use by default for **kneighbors** queries.

weights ***{'uniform', 'distance'}, callable or None, default='uniform'.*** Weight function used in prediction. Possible values:

Metric ***str or callable, default='minkowski'*** Metric to use for distance computation. Default is "minkowski", which results in the standard Euclidean distance when p = 2.

n_jobs ***int, default=None*** The number of parallel jobs to run for neighbour's search

```
              precision    recall  f1-score   support

           0       0.78      0.74      0.76       307
           1       0.90      0.92      0.91       754

    accuracy                           0.86      1061
   macro avg       0.84      0.83      0.83      1061
weighted avg       0.86      0.86      0.86      1061
```

	precision	recall	f1-score	support
0	0.76	0.73	0.74	153
1	0.87	0.88	0.87	303
accuracy			0.83	456
macro avg	0.81	0.81	0.81	456
weighted avg	0.83	0.83	0.83	456

The above table shows the Accuracy, Precision, Recall and the F1 of the Logistic Regression model for both the train and the test data.

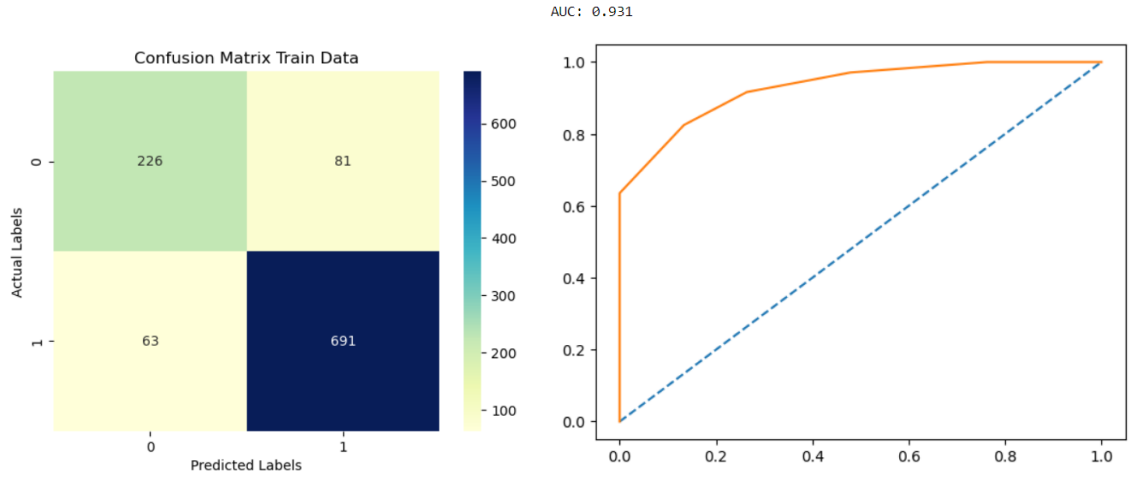
• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 90% for the train and 87% for the test data for classifier 1 and 78% for training and 76% for testing data for classifier 0.

• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 92% for the train data and 88% for the test data for classifier 1 and 74% for training and 73% for testing data for classifier 0.

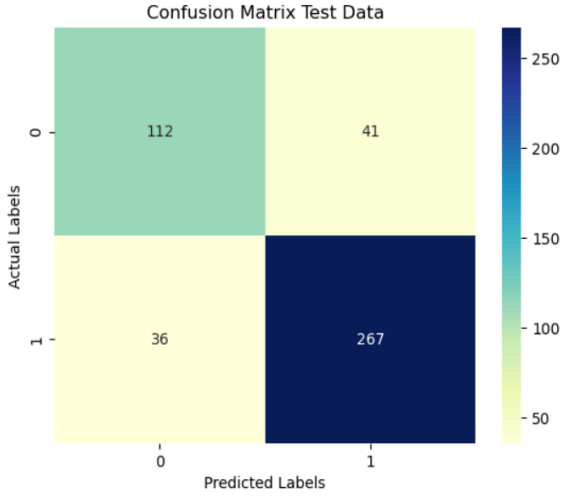
• **F1-score** : Here it is 91% for train data and 87% for test data for classifier 1 and 76% for train data and 74% for test data for classifier 0

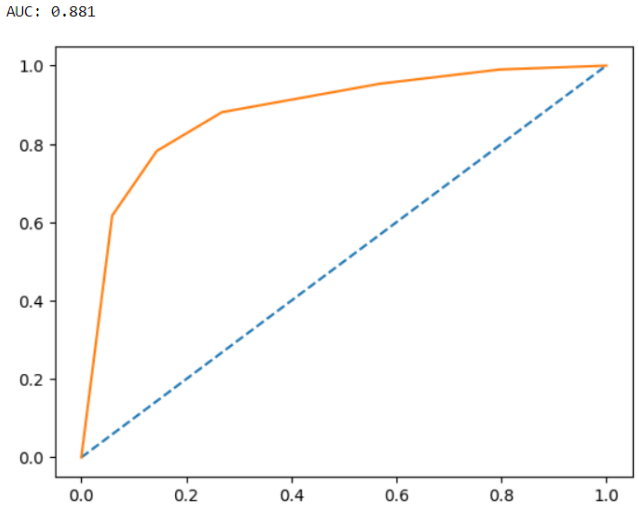
• **Accuracy**: The train data accuracy is 86% and the test data accuracy is 83%.

In this case the model accuracy is slightly different for both training and testing dataset so there is no over or underfitting .Its a good Model



We can from the confusion matrix above see that 226 of 307 of class 0 are precited correctly and 691 of 754 of class 1 are predicted correctly in training data





:

We can from the confusion matrix above see that 112 of 153 of class 0 are predcited correctly and 267 of 303 of class 1 are predicted correctly in testing data

AUC scores for testing is 0.881 and training data is 0.931

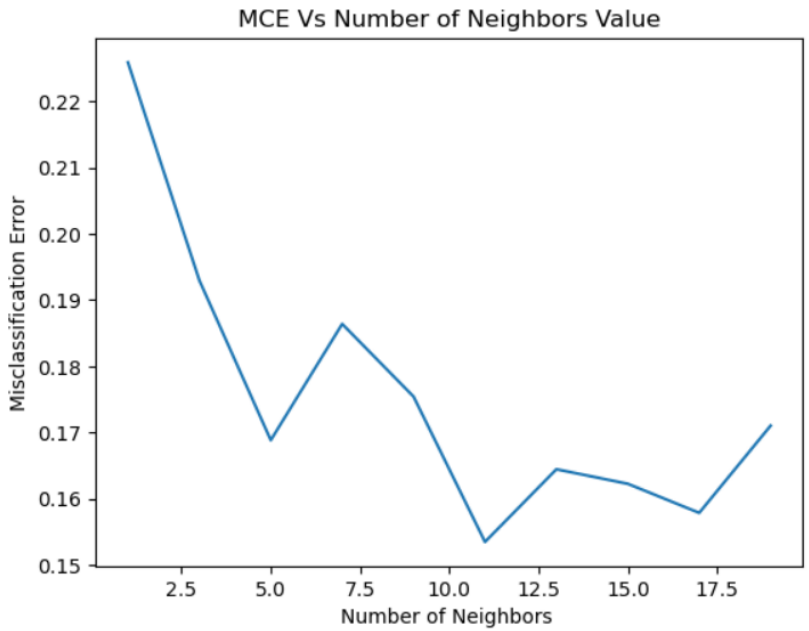
KNN with optimum value of neighbours

To find the best value of n_neighbours) ,we build **KNN** model with different n_neighbours(1,3,5..19) to find which one will give least Misclassification error (MCE) .Then we can compute the MCE(1 - Test accuracy score) for each value of n_neighbours. The model with lowest MCE is the best value for n_neighbours.

After running the model with different values of n neighbours we get the MCE values as follows

```
For n_neighbors = 1 value of MCE is  0.22587719298245612
For n_neighbors = 3 value of MCE is  0.19298245614035092
For n_neighbors = 5 value of MCE is  0.16885964912280704
For n_neighbors = 7 value of MCE is  0.1864035087719298
For n_neighbors = 9 value of MCE is  0.17543859649122806
For n_neighbors = 11 value of MCE is  0.1535087719298246
For n_neighbors = 13 value of MCE is  0.16447368421052633
For n_neighbors = 15 value of MCE is  0.16228070175438591
For n_neighbors = 17 value of MCE is  0.1578947368421053
For n_neighbors = 19 value of MCE is  0.17105263157894735
```

Let's plot the different values of neighbours against MCE



As we can see K=5 and K=11 , there is sharp dip

We saw the model performance at K=5 which was the default parameters , now we can see for K=11

	precision	recall	f1-score	support
0	0.78	0.70	0.74	307
1	0.88	0.92	0.90	754
accuracy			0.86	1061
macro avg	0.83	0.81	0.82	1061
weighted avg	0.85	0.86	0.86	1061

	precision	recall	f1-score	support
0	0.81	0.71	0.76	153
1	0.86	0.92	0.89	303
accuracy			0.85	456
macro avg	0.84	0.81	0.82	456
weighted avg	0.84	0.85	0.84	456

The above table shows the Accuracy, Precision, Recall and the F1 of the Logistic Regression model for both the train and the test data.

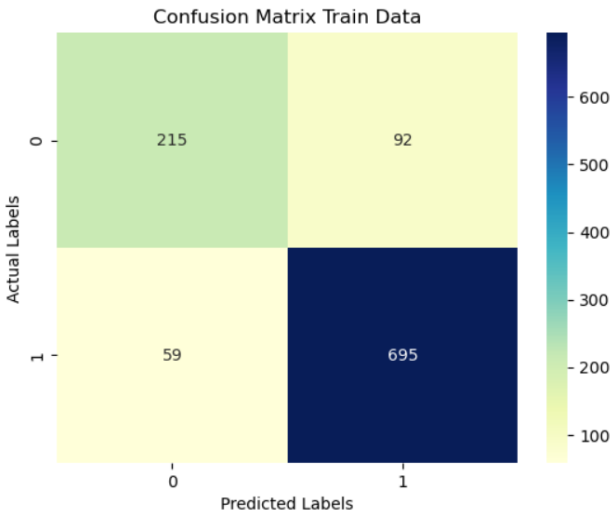
• **Precision** : it is 88% for the train and 86% for the test data for classifier 1 and 78% for training and 81% for testing data for classifier 0.

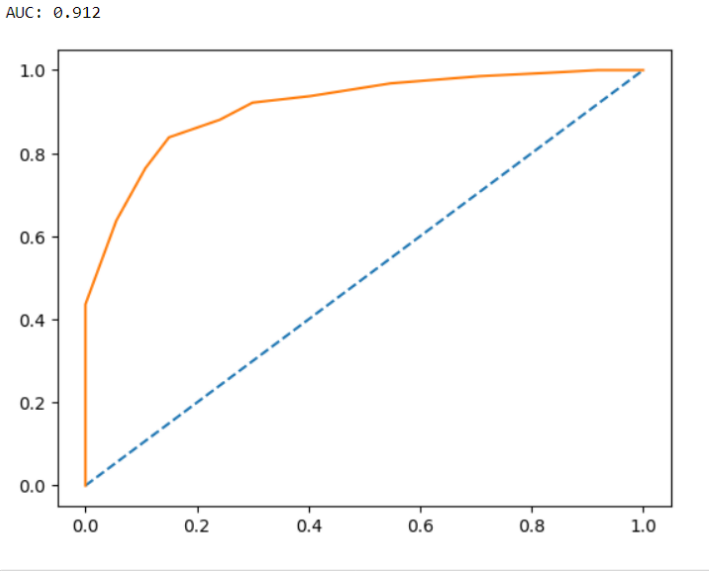
• **Recall** : it is 92% for the train data and 92% for the test data for classifier 1 and 70% for training and 71% for testing data for classifier 0.

• **F1-score** : it is 90% for train data and 89% for test data for classifier 1 and 74% for train data and 76% for test data for classifier 0

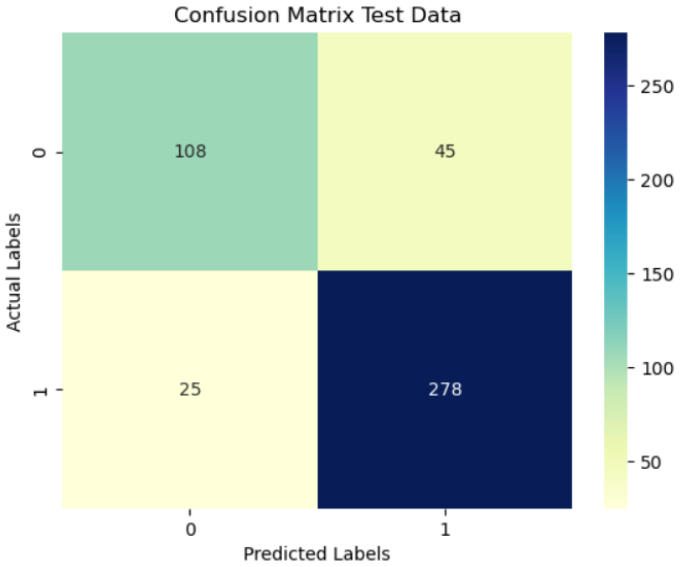
• **Accuracy**: The train data accuracy is 86% and the test data accuracy is 85%.

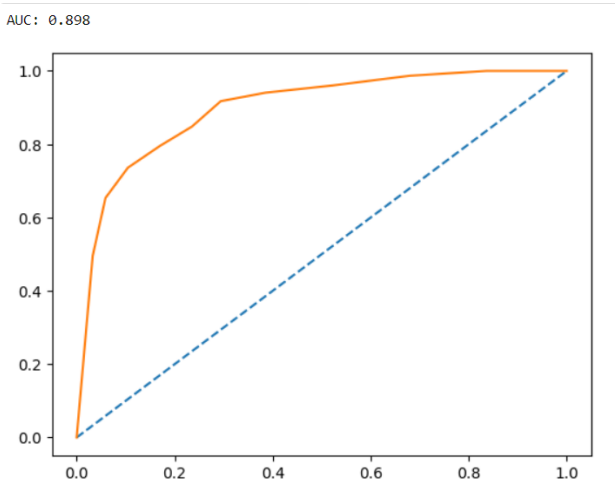
In this case the model accuracy is slightly different for both training and testing dataset so there is no over or underfitting .





We can from the confusion matrix above see that 215 of 307 of class 0 are precited correctly and 695 of 754 of class 1 are predicted correctly in training data





We can from the confusion matrix above see that 108 of 153 of class 0 are predicted correctly and 278 of 303 of class 1 are predicted correctly in testing data

AUC scores for testing is 0.898 and training data is 0.912

Gaussian Naïve Bayes

Let's build Gaussian Naïve Bayes model with default hyper parameters

Parameters	Value
var_smoothing	1e-9

priors : array-like of shape (n_classes,) Prior probabilities of the classes. If specified the priors are not adjusted according to the data.

var_smoothing : float, default=1e-9 Portion of the largest variance of all features that is added to variances for calculation stability.

```
              precision    recall  f1-score   support

           0       0.73      0.69      0.71       307
           1       0.88      0.89      0.88       754

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

              precision    recall  f1-score   support

           0       0.75      0.73      0.74       153
           1       0.87      0.87      0.87       303

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

The above table shows the Accuracy, Precision, Recall and the F1 of the Logistic Regression model for both the train and the test data.

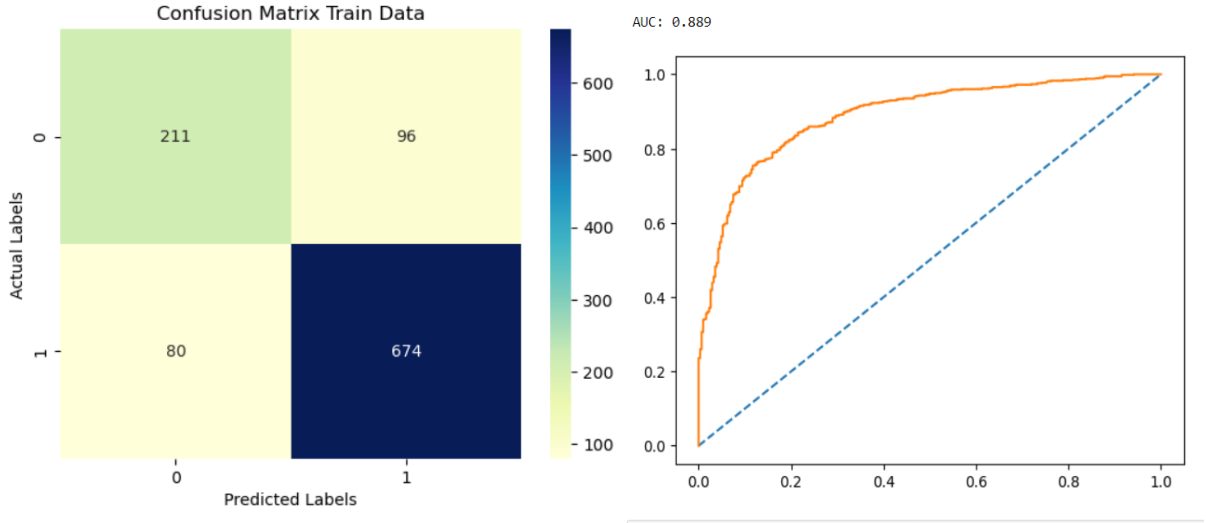
• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 88% for the train and 87% for the test data for classifier 1 and 73% for training and 75% for testing data for classifier 0.

• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 89% for the train data and 87% for the test data for classifier 1 and 69% for training and 73% for testing data for classifier 0.

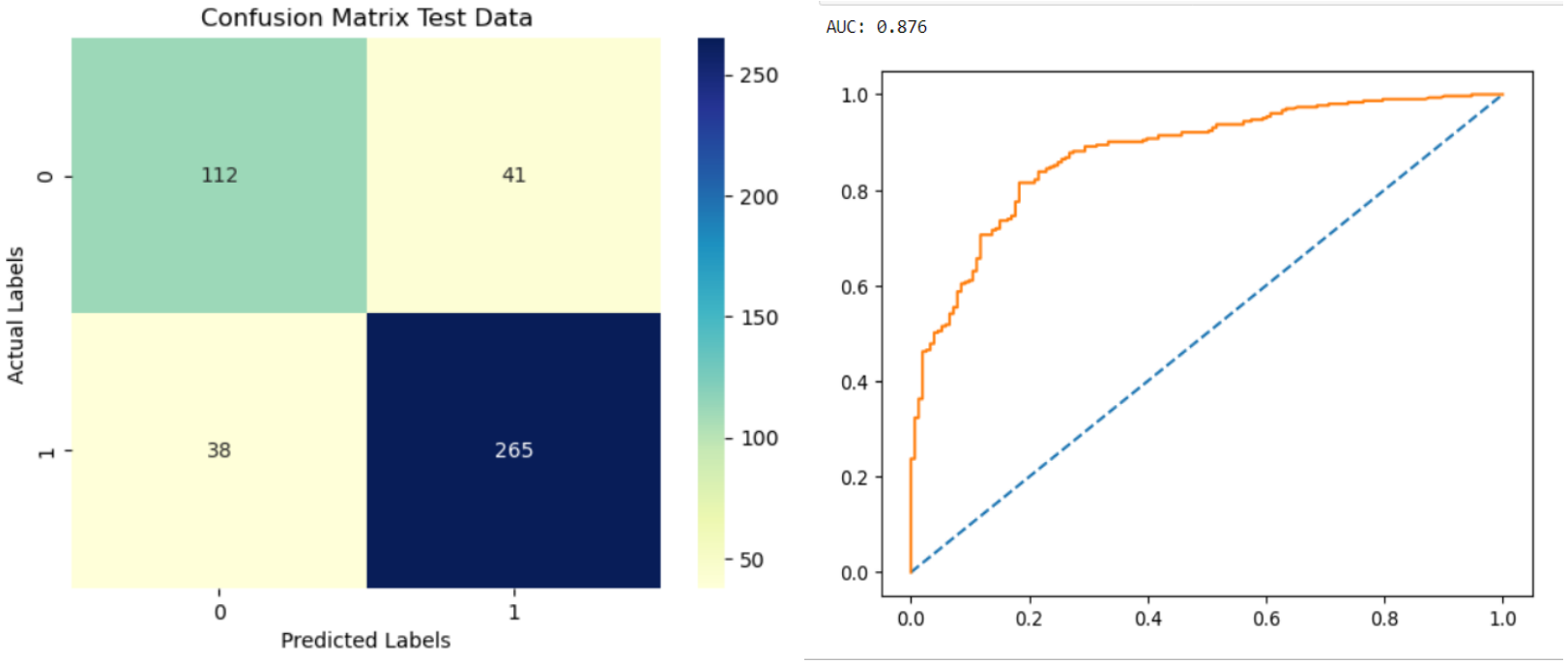
• **F1-score** : Here it is 88% for train data and 87% for test data for classifier 1 and 71% for train data and 74% for test data for classifier 0

• **Accuracy**: The train data accuracy is 83% and the test data accuracy is 83%.

When a model performs very well in the train data and but doesn't perform well in the testing set performs ,it's called overfitting and the inverse is called the underfitting. In this case the model accuracy is same for both training and testing dataset so there is no over or underfitting .



We can from the confusion matrix above see that 211 of 307 of class 0 are precited correctly and 674 of 754 of class 1 are predicted correctly



We can from the confusion matrix above see that 112 of 153 of class 0 are precited correctly and 265 of 303 of class 1 are predicted correctly in testing data

AUC scores for testing 0.876 and training data 0.889 is very close , so there is no overfitting .

1.6) Model Tuning (4 pts) , Bagging (1.5 pts) and Boosting (1.5 pts). Apply grid search on each model (include all models) and make models on best_params. 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

Linear Discriminant Analysis with the best parameters :

Let's build a Linear Discriminant Analysis model with best hyper parameters using GridSearchCV function

GridSearchCV is the process of performing hyperparameter tuning in order to determine the optimal values for a given model. The performance of a model significantly depends on the value of hyperparameters. we use GridSearchCV to automate the tuning of hyperparameters. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

The param grid that was chosen was 'solver':['svd', 'lsqr', 'eigen'] and 'tol':[0.0001,0.00001]

7

GridSearchCV best estimator model also comes with same parameters that were used for default parameters 'solver':'svd' and 'tol':0.0001.

	precision	recall	f1-score	support
0	0.74	0.66	0.70	307
1	0.87	0.91	0.89	754
accuracy			0.84	1061
macro avg	0.81	0.78	0.79	1061
weighted avg	0.83	0.84	0.83	1061

	precision	recall	f1-score	support
0	0.77	0.75	0.76	153
1	0.87	0.89	0.88	303
accuracy			0.84	456
macro avg	0.82	0.82	0.82	456
weighted avg	0.84	0.84	0.84	456

The above table shows the Accuracy, Precision, Recall and the F1 of the LDA model for both the train and the test data.

• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 87% for both the train and the test data for classifier 1 and 74% for training and 77% for testing data for classifier 0.

• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 91% for the train data and 89% for the test data for classifier 1 and 66% for training and 75% for testing data for classifier 0.

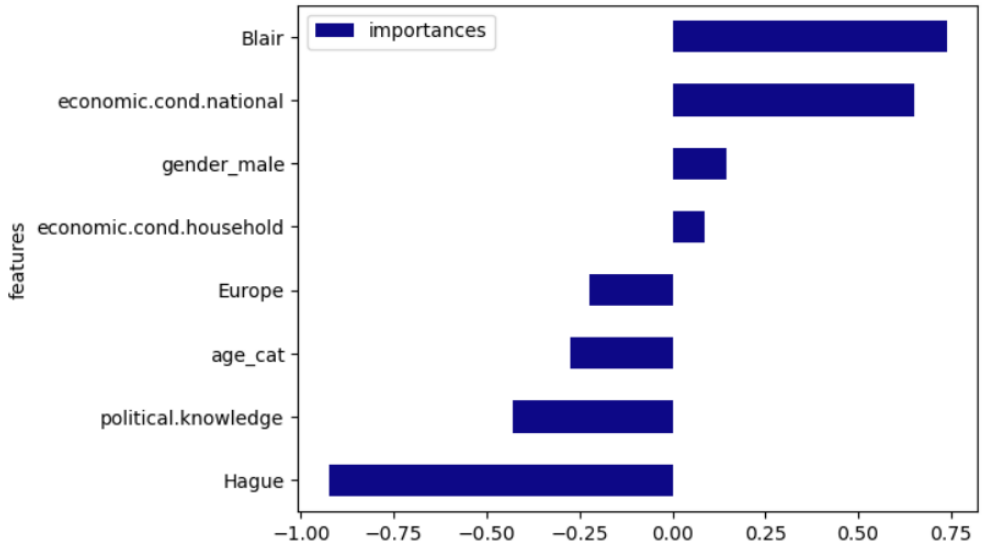
• **F1-score** : It combines the precision and recall of a classifier into a single metric by taking their harmonic mean. It is primarily used to compare the performance of two classifiers. Suppose that classifier A has a higher recall, and classifier B has higher precision. In this case, the F1-scores for both the classifiers can be used to determine which one produces better results. The greater the F1 Score, the better is the performance of our model. Here it is 89% for train data and 88% for test data for classifier 1 and 70% for train data and 76% for test data for classifier 0

• **Accuracy**: The train data accuracy is 84% and the test data accuracy is 84%.

In this case the model accuracy is same for both training and testing dataset so there is no over or underfitting . So its a good Model

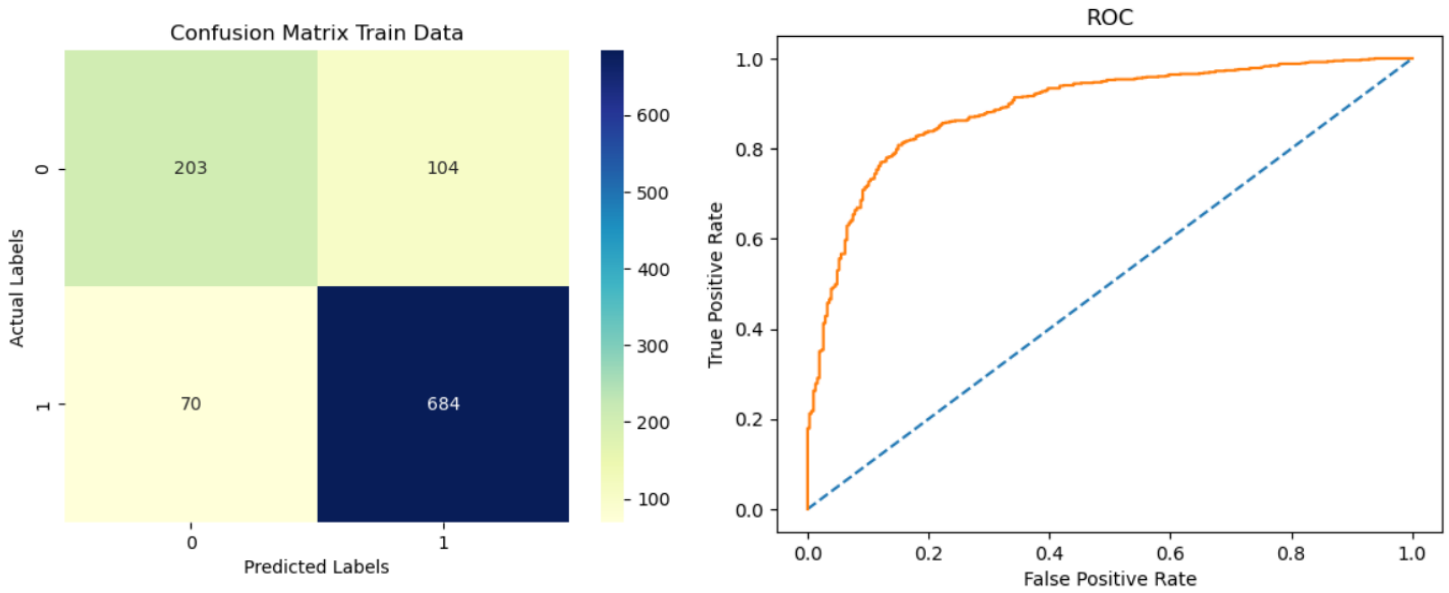
Since Class 0 is not represented less compared to Class 1 recall of Class 1 is more compared to Class 0

Feature Importance

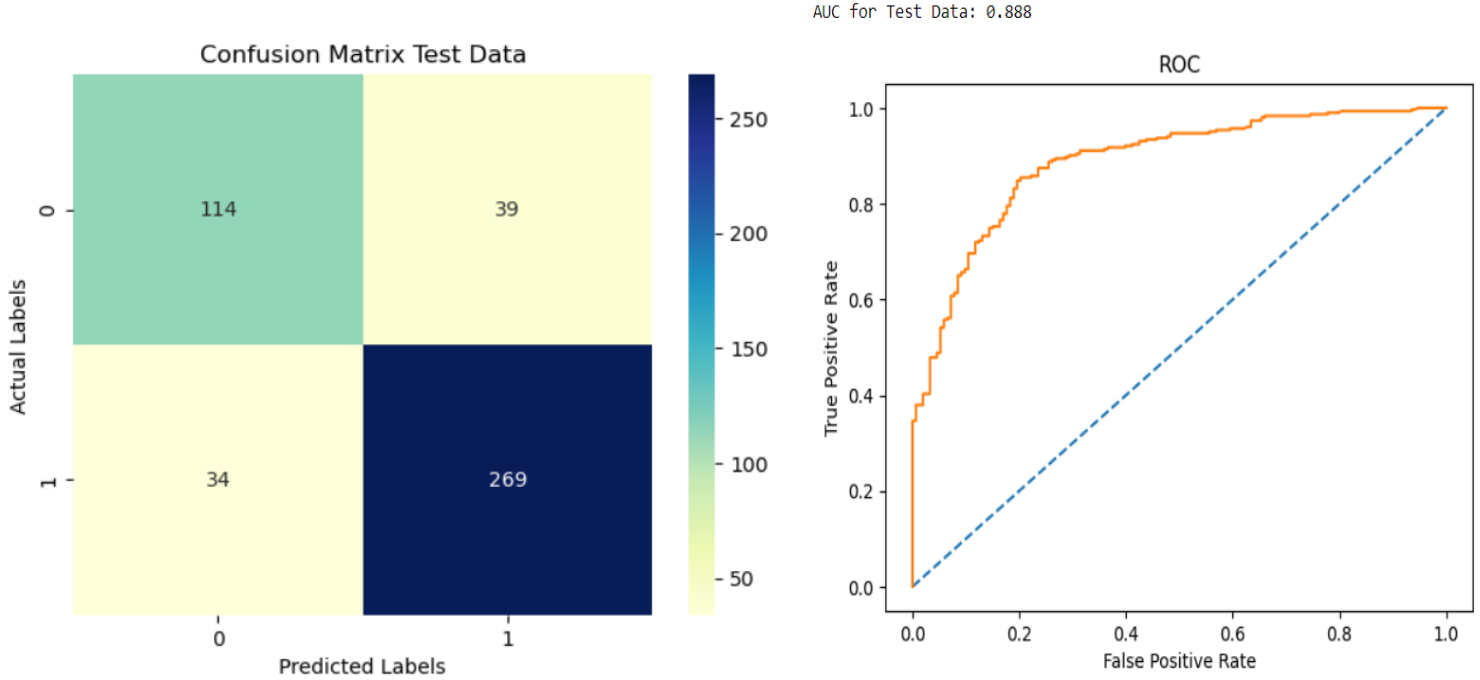


As per the Feature Importance we can see that Blair, Hague, economic condition national and Political knowledge are some of the most important features in the dataset

AUC for Training Data: 0.890



We can from the confusion matrix above see that 203 of 307 of class 0 are precited correctly and 684 of 754 of class 1 are predicted correctly



We can from the confusion matrix above see that 114 of 153 of class 0 are precited correctly and 269 of 303 of class 1 are predicted correctly in testing data

AUC scores for testing 0.888 and training data 0.890 is very close , so there is no overfitting

Logistic Regression with the best parameters :

Let's build a **Logistic Regression** model with best hyper parameters using GridSearchCV function

GridSearchCV is the process of performing hyperparameter tuning in order to determine the optimal values for a given model. The performance of a model significantly depends on the value of hyperparameters. we use GridSearchCV to automate the tuning of hyperparameters. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

The param grid that was chosen was

'penalty':['l1','l2','none'],'solver':['sag','lbfgs','newton-cg','liblinear','saga'], 'tol':[0.0001,0.00001],'max_iter':[1000,10000,100000]

GridSearchCV best estimator model comes up with parameters {'max_iter': 1000, 'penalty': 'l1', 'solver': 'liblinear', 'tol': 0.0001}

Let's build Logistic Regression model with following hyper parameters

Parameters	**Value**
solver	'liblinear'
tol	0.0001
max_iter	1000
penalty	l1

	precision	recall	f1-score	support
0	0.76	0.64	0.70	307
1	0.86	0.92	0.89	754
accuracy			0.84	1061
macro avg	0.81	0.78	0.79	1061
weighted avg	0.83	0.84	0.83	1061

	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

The above table shows the Accuracy, Precision, Recall and the F1 of the Logistic Regression model for both the train and the test data.

• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 86% for both the train and the test data for classifier 1 and 76% for training and 76% for testing data for classifier 0.

• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 92% for the train data and 88% for the test data for classifier 1 and 64% for training and 73% for testing data for classifier 0.

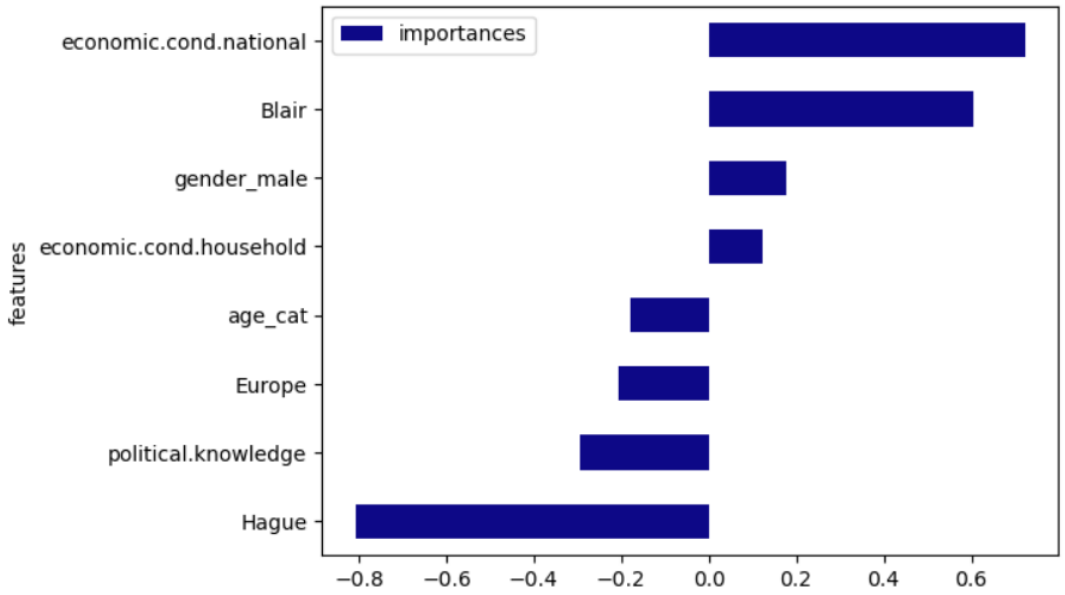
• **F1-score** : Here it is 89% for train data and 87% for test data for classifier 1 and 70% for train data and 74% for test data for classifier 0

• **Accuracy**: The train data accuracy is 84% and the test data accuracy is 83%. In this case the model accuracy is almost same for training and testing dataset so there is no over or underfitting .Accuracy has not changed in testing set after hyperparameter tuning .

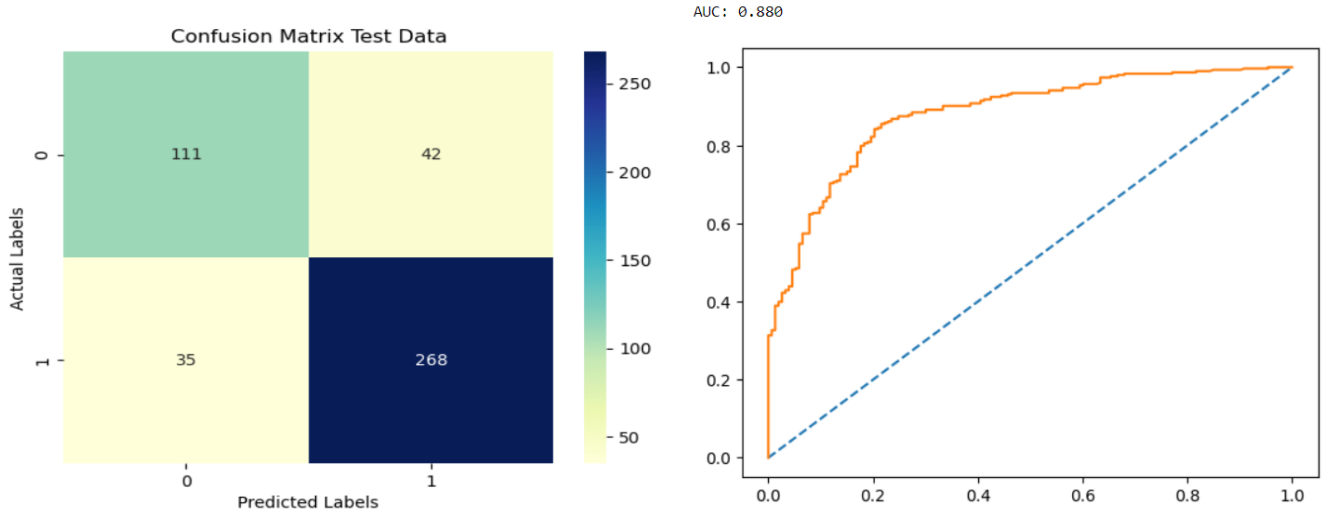
In this case the model accuracy is almost same for both training and testing dataset so there is no over or underfitting . So its a good Model

Since Class 0 is not represented less compared to Class 1 recall of Class 1 is more compared to Class 0

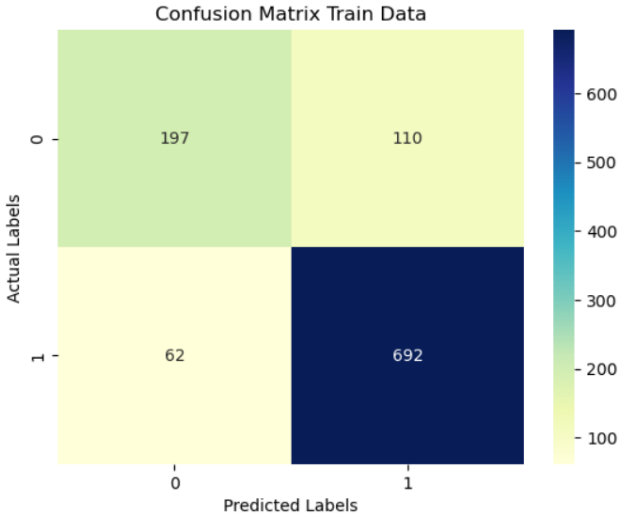
Feature Importance

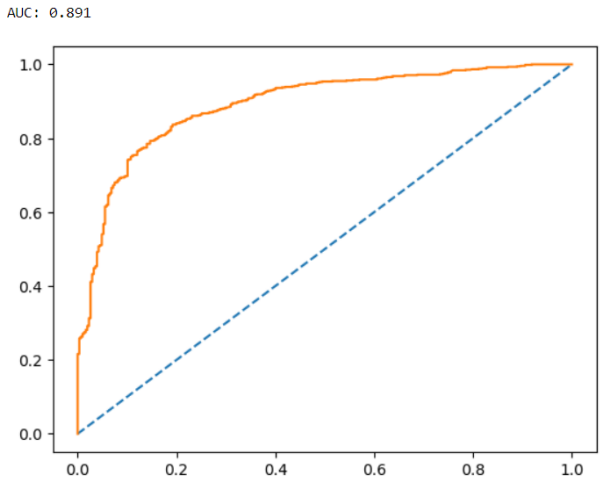


As per the Feature Importance we can see that Blair, Hague, economic condition national and Political knowledge are some of the most important features in the dataset



We can from the confusion matrix above see that 111 of 153 of class 0 are precited correctly and 268 of 303 of class 1 are predicted correctly in testing data





We can from the confusion matrix above see that 197 of 307 of class 0 are precited correctly and 692 of 754 of class 1 are predicted correctly

AUC scores for testing 0.888 and training data 0.891 is very close , so there is no overfitting

KNN GridSearchCV

GridSearchCV is the process of performing hyperparameter tuning in order to determine the optimal values for a given model. The performance of a model significantly depends on the value of hyperparameters. we use GridSearchCV to automate the tuning of hyperparameters. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

The param grid that was chosen was

'weights':['uniform', 'distance'],'metric':['euclidean', 'manhattan','minkowski'], 'n_neighbors':[1,3, 5, 7, 9, 11, 13,15,17,19]

GridSearchCV best estimator model comes up with parameters { 'metric': 'manhattan', 'n_neighbors': 19, 'weights': unform}

Let's build KNN model with following hyper parameters

Parameters	**Value**
'metric'	'manhattan'
'n_neighbors	15
'weights'	`uniform`

	precision	recall	f1-score	support
0	0.75	0.69	0.72	307
1	0.88	0.91	0.89	754
accuracy			0.85	1061
macro avg	0.82	0.80	0.81	1061
weighted avg	0.84	0.85	0.84	1061

```
[ 52 271]]
              precision    recall  f1-score   support

           0       0.77      0.70      0.73       153
           1       0.85      0.89      0.87       303

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

The above table shows the Accuracy, Precision, Recall and the F1 of the Logistic Regression model for both the train and the test data.

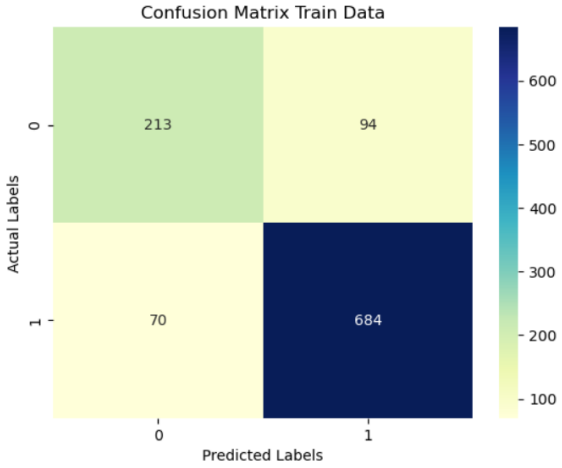
• **Precision** : it is 88% for the train and 85% for the test data for classifier 1 and 75% for training and 77% for testing data for classifier 0.

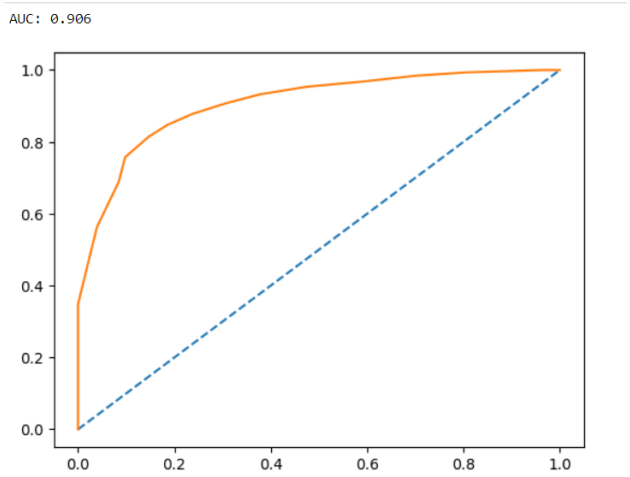
• **Recall** : it is 91% for the train data and 89 % for the test data for classifier 1 and 69% for training and 70% for testing data for classifier 0.

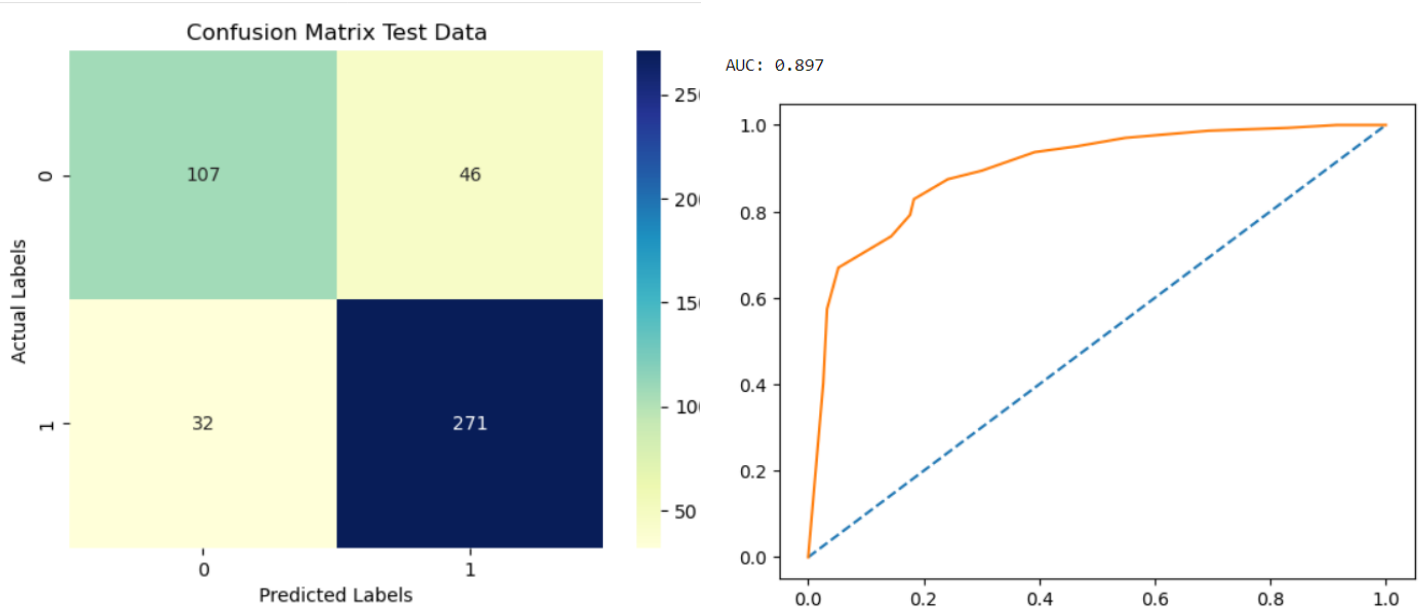
• **F1-score** : it is 89% for train data and 87% for test data for classifier 1 and 72% for train data and 73% for test data for classifier 0

• **Accuracy**: The train data accuracy is 85% and the test data accuracy is 83%.

In this case the model accuracy of training set is 85 % and of testing dataset is 83% so there is no over- fitting and is a suitable model


Confusion Matrix Train Data
213
94
70
684
Actual Labels
Predicted Labels
0
1
600
500
400
300
200
100


AUC: 0.906
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0



We can from the confusion matrix above see that 213 of 307 of class 0 are precited correctly and 684 of 754 of class 1 are predicted correctly

We can from the confusion matrix above see that 107 of 153 of class 0 are precited correctly and 271 of 303 of class 1 are predicted correctly in testing data

AUC scores for testing 0.897 and training data 0.906 is very close , so there is no overfitting

KNN with SMOTE

SMOTE adds synthetic data to the training sample to balance the data belonging to minority class .

As we can see 447 new observations are added to the minority class to balance the data . There are 754 records each 0 and 1 class .

	precision	recall	f1-score	support
0	0.84	0.94	0.89	754
1	0.93	0.82	0.87	754
accuracy			0.88	1508
macro avg	0.89	0.88	0.88	1508
weighted avg	0.89	0.88	0.88	1508

	precision	recall	f1-score	support
0	0.69	0.81	0.74	153
1	0.89	0.81	0.85	303
accuracy			0.81	456
macro avg	0.79	0.81	0.80	456
weighted avg	0.82	0.81	0.81	456

The above table shows the Accuracy, Precision, Recall and the F1 of the Logistic Regression model for both the train and the test data.

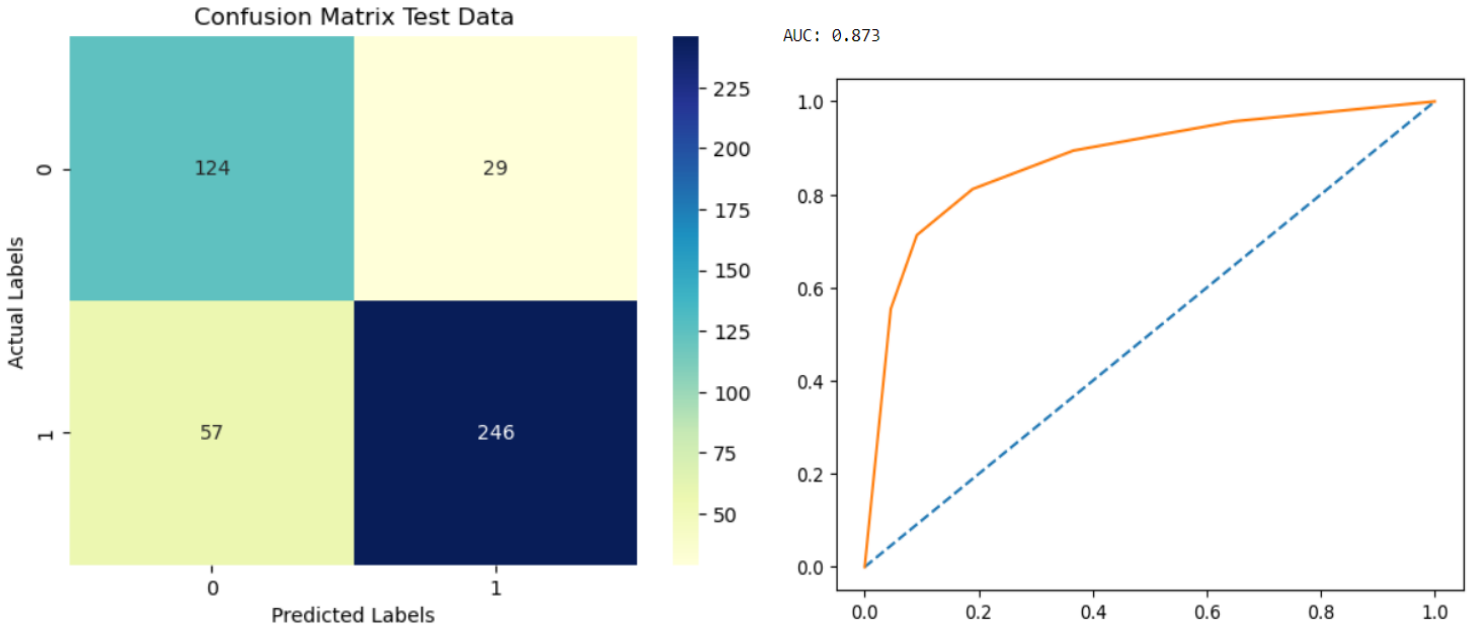
• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 93% for the train and 89% for the test data for classifier 1 and 84% for training and 69% for testing data for classifier 0.

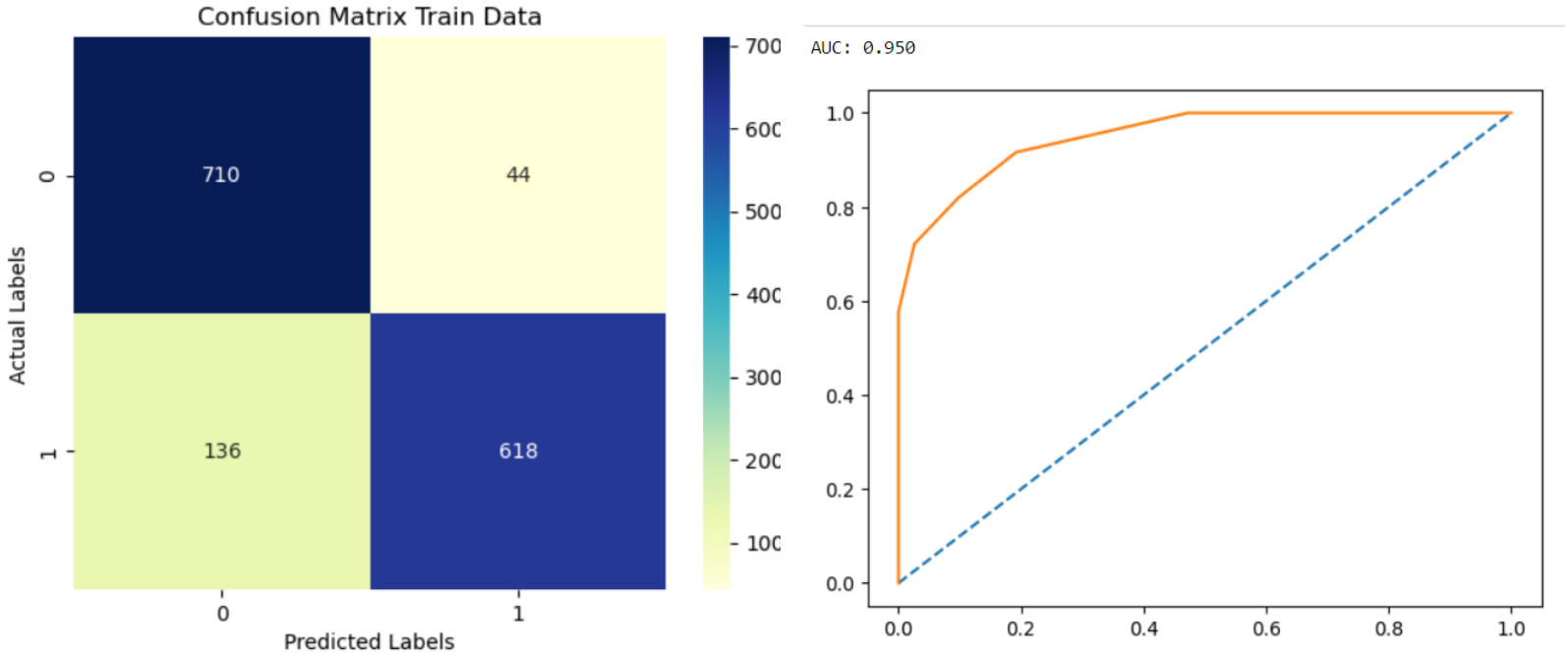
• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 82% for the train data and 81% for the test data for classifier 1 and 94% for training and 81% for testing data for classifier 0.

• **F1-score** : Here it is 87% for train data and 85% for test data for classifier 1 and 89% for train data and 74% for test data for classifier 0

• **Accuracy**: The train data accuracy is 88% and the test data accuracy is 81%.

In this case the model accuracy of training set is 88% and of testing dataset is 81% so there is no over- fitting and is a suitable model .


Confusion Matrix Test Data
124
29
57
246
0
1
Actual Labels
Predicted Labels
225
200
175
150
125
100
75
50
AUC: 0.873
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0



We can from the confusion matrix above see that 710 of 754 of class 0 are precited correctly and 618 of 754 of class 1 are predicted correctly in training data

We can from the confusion matrix above see that 124 of 153 of class 0 are precited correctly and 246 of 303 of class 1 are predicted correctly in testing data

AUC scores for testing 0.873 and training data 0.95, so there is no overfitting.

Naive Bayes with SMOTE

SMOTE adds synthetic data to the training sample to balance the data belonging to minority class .

As we can see 447 new observations are added to the minority class to balance the data . There are 754 records each 0 and 1 class .

	precision	recall	f1-score	support
0	0.82	0.83	0.82	754
1	0.82	0.81	0.82	754
accuracy			0.82	1508
macro avg	0.82	0.82	0.82	1508
weighted avg	0.82	0.82	0.82	1508

	precision	recall	f1-score	support
0	0.67	0.81	0.73	153
1	0.89	0.80	0.84	303
accuracy			0.80	456
macro avg	0.78	0.80	0.79	456
weighted avg	0.82	0.80	0.80	456

The above table shows the Accuracy, Precision, Recall and the F1 of the Logistic Regression model for both the train and the test data.

• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 82% for the train and 89% for the test data for classifier 1 and 82% for training and 67% for testing data for classifier 0.

• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 81% for the train data and 80% for the test data for classifier 1 and 83% for training and 81% for testing data for classifier 0.

• **F1-score** : Here it is 82% for train data and 84% for test data for classifier 1 and 82% for train data and 73% for test data for classifier 0

• **Accuracy**: The train data accuracy is 82% and the test data accuracy is 80%.

When a model performs very well in the train data and but doesn't perform well in the testing set performs ,it's called overfitting and the inverse is called the underfitting. In this case the model accuracy is slightly different for both training and testing dataset so there is no over or underfitting .

Bagging

Bagging is an ensemble algorithm that fits multiple models on different subsets of a training dataset, then combines the predictions from all models.

Random forest is an extension of bagging that also randomly selects subsets of features used in each data sample. Both bagging and random forests have proven effective on a wide range of different predictive modelling problems.

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Bagging with Decision Tree classifier

```
0.8092105263157895
[[100  53]
 [ 34 269]]
              precision    recall  f1-score   support

           0       0.75      0.65      0.70       153
           1       0.84      0.89      0.86       303

    accuracy                           0.81       456
   macro avg       0.79      0.77      0.78       456
weighted avg       0.81      0.81      0.81       456

0.9905749293119699
[[301   6]
 [  4 750]]
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       307
           1       0.99      0.99      0.99       754

    accuracy                           0.99      1061
   macro avg       0.99      0.99      0.99      1061
weighted avg       0.99      0.99      0.99      1061
```

The above table shows the Accuracy, Precision, Recall and the F1 of the Bagging with Decision Tree classifier model for both the train and the test data.

• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 99% for the train and 84% for the test data for classifier 1 and 99% for training and 75% for testing data for classifier 0.

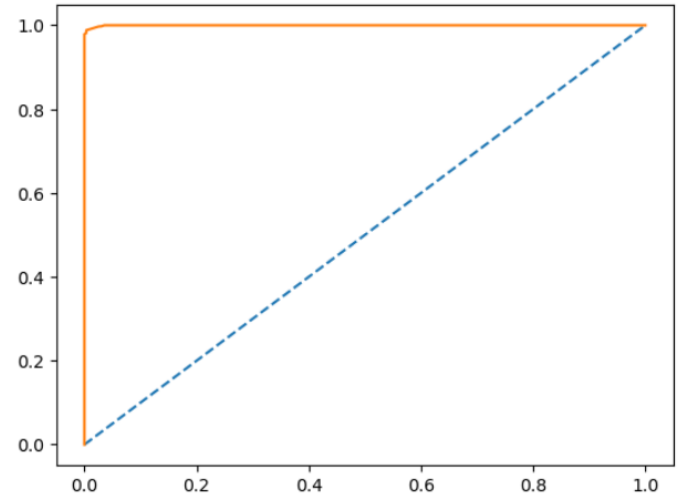
• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 99% for the train data and 89% for the test data for classifier 1 and 98% for training and 65% for testing data for classifier 0.

• **F1-score** : Here it is 99% for train data and 86% for test data for classifier 1 and 98% for train data and 70% for test data for classifier 0

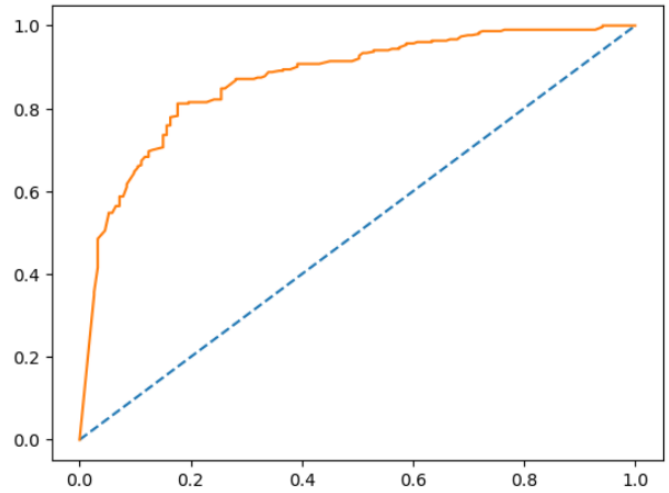
• **Accuracy**: The train data accuracy is 99% and the test data accuracy is 81%.

When a model performs very well in the train data and but doesn't perform well in the testing set performs ,it's called overfitting and the inverse is called the underfitting. In this case the model accuracy for training is very high compared to testing dataset so there is overfitting.

```
AUC: 1.000
```



```
AUC: 0.870
```



Bagging with Random Forest classifier

```
0.8333333333333334
[[101  52]
 [ 24 279]]
              precision    recall  f1-score   support

           0       0.81      0.66      0.73       153
           1       0.84      0.92      0.88       303

    accuracy                           0.83       456
   macro avg       0.83      0.79      0.80       456
weighted avg       0.83      0.83      0.83       456

0.9622997172478793
[[274  33]
 [  7 747]]
              precision    recall  f1-score   support

           0       0.98      0.89      0.93       307
           1       0.96      0.99      0.97       754

    accuracy                           0.96      1061
   macro avg       0.97      0.94      0.95      1061
weighted avg       0.96      0.96      0.96      1061
```

The above table shows the Accuracy, Precision, Recall and the F1 of the Bagging with Decision Tree classifier model for both the train and the test data.

• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 96% for the train and 84% for the test data for classifier 1 and 98% for training and 81% for testing data for classifier 0.

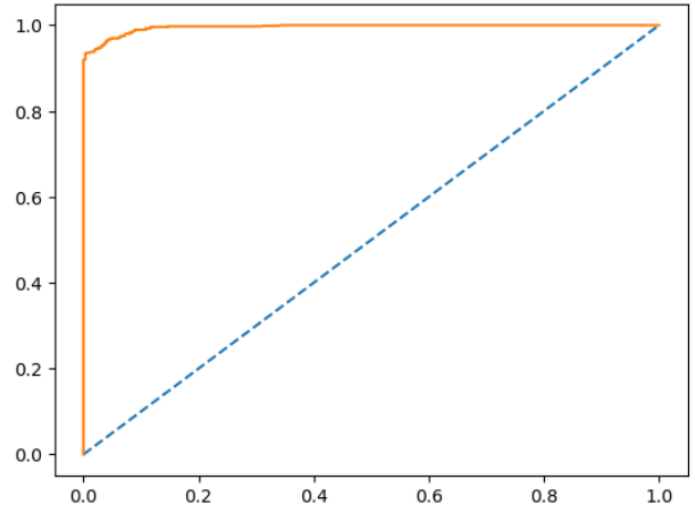
• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 99% for the train data and 92% for the test data for classifier 1 and 89% for training and 66% for testing data for classifier 0.

• **F1-score** : Here it is 97% for train data and 88% for test data for classifier 1 and 93% for train data and 73% for test data for classifier 0

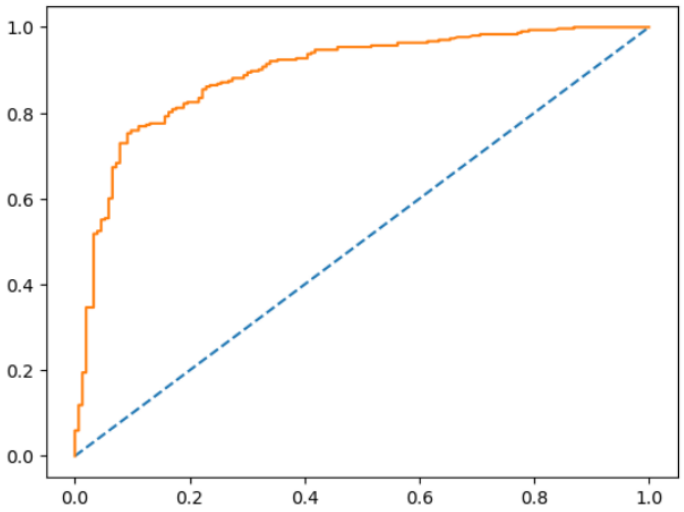
• **Accuracy**: The train data accuracy is 96% and the test data accuracy is 83%.

When a model performs very well in the train data and but doesn't perform well in the testing set performs ,it's called overfitting and the inverse is called the underfitting. In this case the model accuracy for training is very high compared to testing dataset so there is overfitting.

```
AUC: 0.996
```



```
AUC: 0.892
```



Ada Boosting

```
0.8482563619227145
[[212  95]
 [ 66 688]]
              precision    recall  f1-score   support

           0       0.76      0.69      0.72       307
           1       0.88      0.91      0.90       754

    accuracy                           0.85      1061
   macro avg       0.82      0.80      0.81      1061
weighted avg       0.85      0.85      0.85      1061
```

```
0.8245614035087719
[[105  48]
 [ 32 271]]
              precision    recall  f1-score   support

           0       0.77      0.69      0.72       153
           1       0.85      0.89      0.87       303

    accuracy                           0.82       456
   macro avg       0.81      0.79      0.80       456
weighted avg       0.82      0.82      0.82       456
```

The above table shows the Accuracy, Precision, Recall and the F1 of the Bagging with Decision Tree classifier model for both the train and the test data.

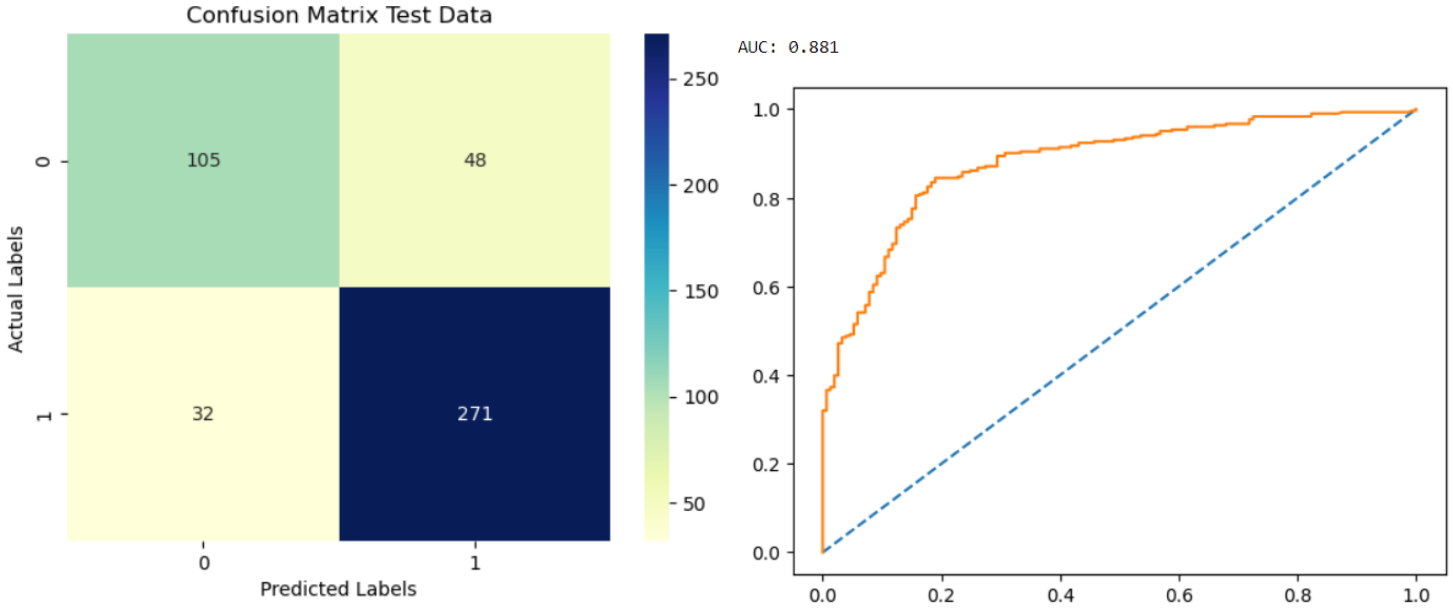
• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 88% for the train and 85% for the test data for classifier 1 and 76% for training and 77% for testing data for classifier 0.

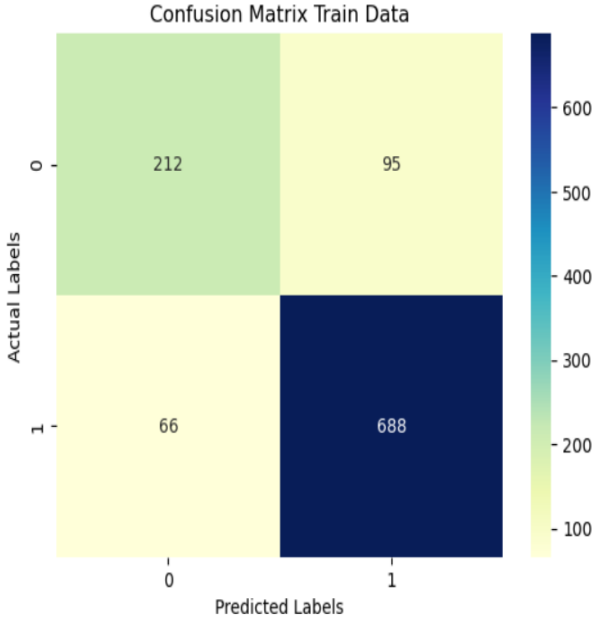
• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 91% for the train data and 89% for the test data for classifier 1 and 69% for training and 69% for testing data for classifier 0.

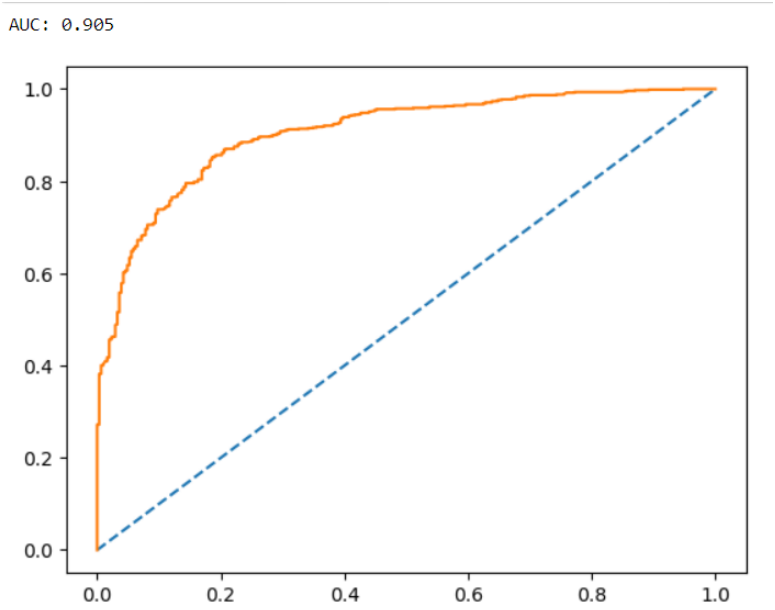
• **F1-score** : Here it is 90% for train data and 87% for test data for classifier 1 and 72% for train data and 72% for test data for classifier 0

• **Accuracy**: The train data accuracy is 85% and the test data accuracy is 82%.

When a model performs very well in the train data and but doesn't perform well in the testing set performs ,it's called overfitting and the inverse is called the underfitting. In this case the model accuracy for training similar to testing dataset so there is no overfitting.


Confusion Matrix Test Data
105
48
32
271
0
1
Actual Labels
Predicted Labels
250
200
150
100
50
AUC: 0.881
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0





Gradient Boosting

0.8821860165990231

	precision	recall	f1-score	support
0	0.82	0.77	0.79	307
1	0.91	0.93	0.92	754
accuracy			0.88	1061
macro avg	0.86	0.85	0.85	1061
weighted avg	0.88	0.88	0.88	1061

	precision	recall	f1-score	support
0	0.81	0.66	0.73	153
1	0.84	0.92	0.88	303
accuracy			0.83	456
macro avg	0.83	0.79	0.80	456
weighted avg	0.83	0.83	0.83	456

The above table shows the Accuracy, Precision, Recall and the F1 of the Bagging with Decision Tree classifier model for both the train and the test data.

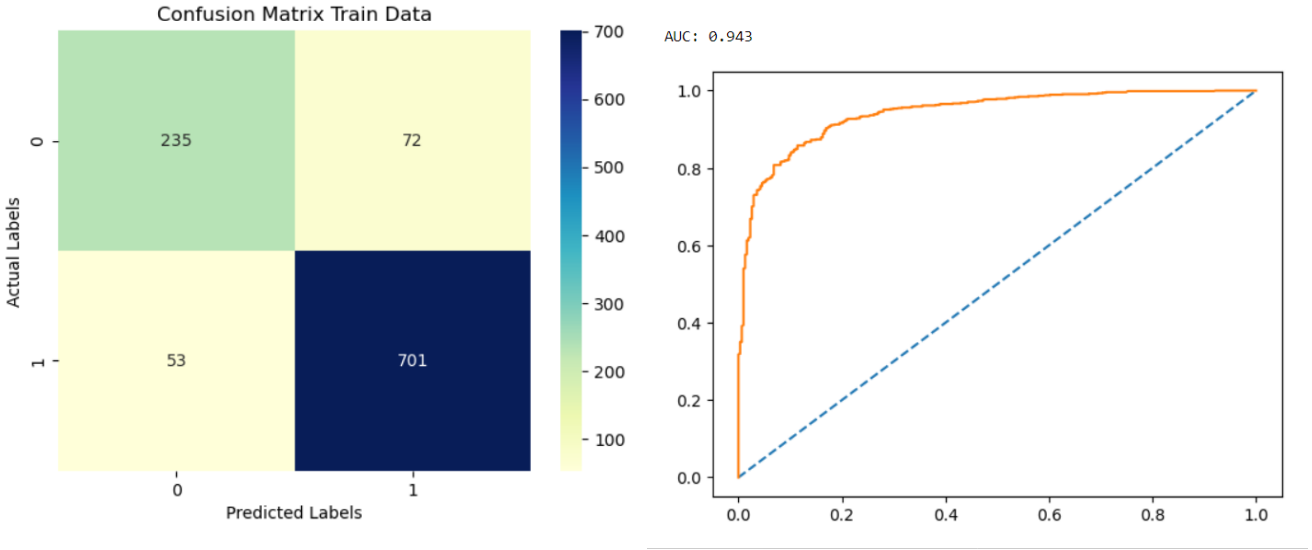
• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 91% for the train and 84% for the test data for classifier 1 and 82% for training and 81% for testing data for classifier 0.

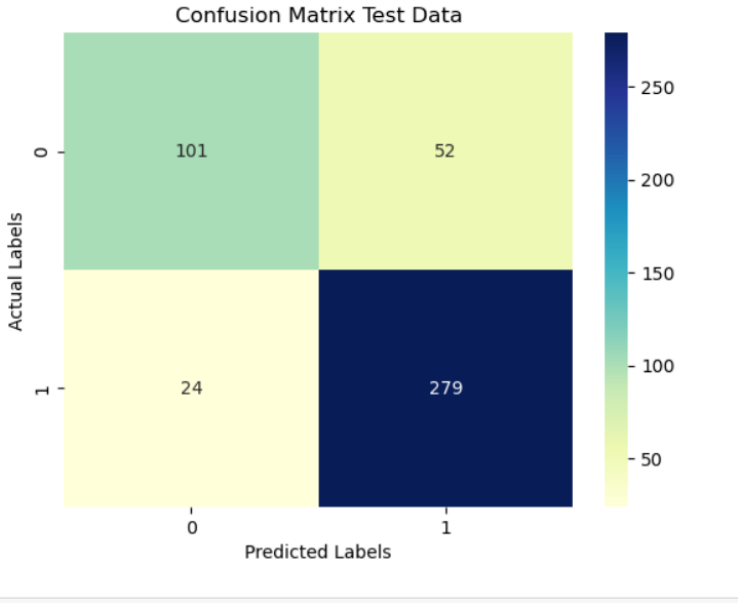
• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 93% for the train data and 92% for the test data for classifier 1 and 76% for training and 66% for testing data for classifier 0.

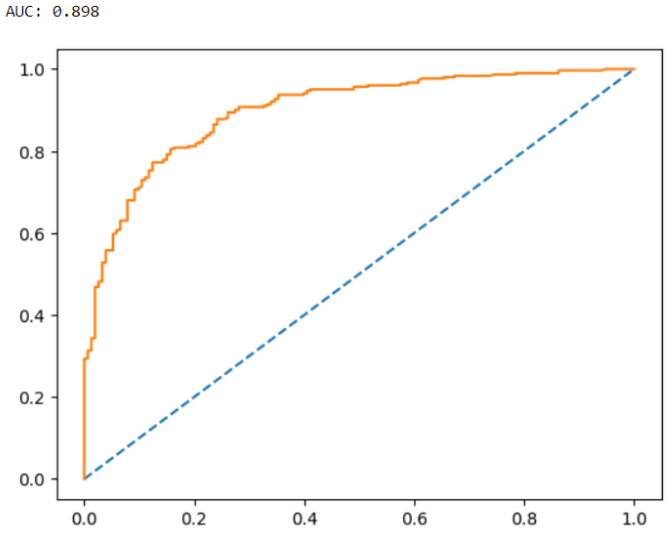
• **F1-score** : Here it is 92% for train data and 88% for test data for classifier 1 and 79% for train data and 73% for test data for classifier 0

• **Accuracy**: The train data accuracy is 88% and the test data accuracy is 83%.

When a model performs very well in the train data and but doesn't perform well in the testing set performs ,it's called overfitting and the inverse is called the underfitting. In this case the model accuracy for training sighlty different from testing dataset so there is no overfitting or underfitting


Confusion Matrix Train Data
235
72
53
701
0
1
Actual Labels
Predicted Labels
700
600
500
400
300
200
100
AUC: 0.943
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Confusion Matrix Test Data
101
52
24
279
0
1
Actual Labels
Predicted Labels
250
200
150
100
50


AUC: 0.898
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

1.7 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model, classification report (4 pts) Final Model - Compare and comment on all models on the basis of the performance metrics in a structured tabular manner. Describe on which model is best/optimized, After comparison which model suits the best for the problem in hand on the basis of different measures. Comment on the final model.(3 pts)

Linear Discriminant Analysis with the best parameters :

Let's build a Linear Discriminant Analysis model with best hyper parameters using GridSearchCV function

GridSearchCV is the process of performing hyperparameter tuning in order to determine the optimal values for a given model. The performance of a model significantly depends on the value of hyperparameters. we use GridSearchCV to automate the tuning of hyperparameters. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

The param grid that was chosen was 'solver':['svd', 'lsqr', 'eigen'] and 'tol':[0.0001,0.00001]

GridSearchCV best estimator model also comes with same parameters that were used for default parameters 'solver':'svd' and 'tol':0.0001.

	precision	recall	f1-score	support
0	0.74	0.66	0.70	307
1	0.87	0.91	0.89	754
accuracy			0.84	1061
macro avg	0.81	0.78	0.79	1061
weighted avg	0.83	0.84	0.83	1061

	precision	recall	f1-score	support
0	0.77	0.75	0.76	153
1	0.87	0.89	0.88	303
accuracy			0.84	456
macro avg	0.82	0.82	0.82	456
weighted avg	0.84	0.84	0.84	456

The above table shows the Accuracy, Precision, Recall and the F1 of the LDA model for both the train and the test data.

• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 87% for both the train and the test data for classifier 1 and 74% for training and 77% for testing data for classifier 0.

• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 91% for the train data and 89% for the test data for classifier 1 and 66% for training and 75% for testing data for classifier 0.

• **F1-score** : It combines the precision and recall of a classifier into a single metric by taking their harmonic mean. It is primarily used to compare the performance of two classifiers. Suppose that classifier A has a higher recall, and classifier B has higher precision. In this case, the F1-scores for both the classifiers can be used to determine which one produces better results. The greater the F1 Score, the better is the performance of our model. Here it is 89% for train data and 88% for test data for classifier 1 and 70% for train data and 76% for test data for classifier 0

• **Accuracy**: The train data accuracy is 84% and the test data accuracy is 84%.

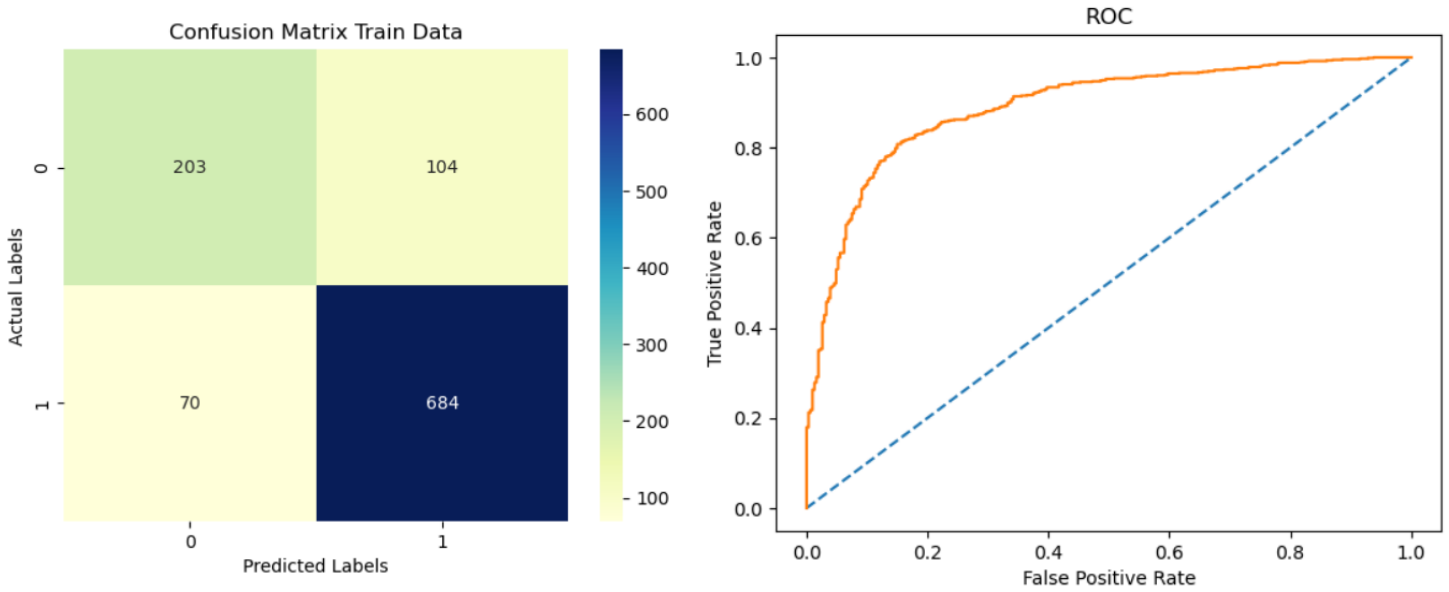
In this case the model accuracy is same for both training and testing dataset so there is no over or underfitting . So its a good Model

Since Class 0 is not represented less compared to Class 1 recall of Class 1 is more compared to Class 0

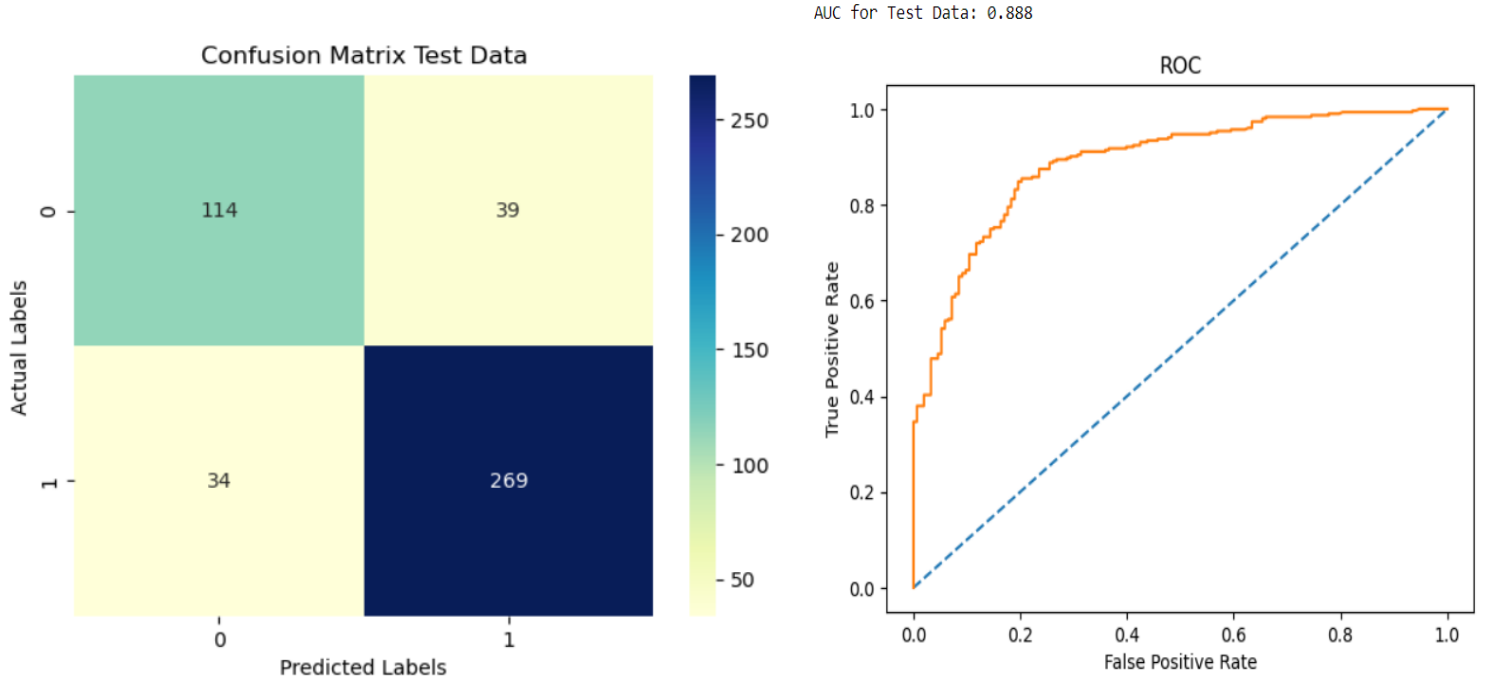
Feature Importance

As per the Feature Importance we can see that Blair, Hague, economic condition national and Political knowledge are some of the most important features in the dataset

AUC for Training Data: 0.890



We can from the confusion matrix above see that 203 of 307 of class 0 are precited correctly and 684 of 754 of class 1 are predicted correctly



We can from the confusion matrix above see that 114 of 153 of class 0 are precited correctly and 269 of 303 of class 1 are predicted correctly in testing data

AUC scores for testing 0.888 and training data 0.890 is very close , so there is no overfitting

Logistic Regression with the best parameters :

Let's build a **Logistic Regression** model with best hyper parameters using GridSearchCV function

GridSearchCV is the process of performing hyperparameter tuning in order to determine the optimal values for a given model. The performance of a model significantly depends on the value of hyperparameters. we use GridSearchCV to automate the tuning of hyperparameters. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

The param grid that was chosen was

'penalty':['l1','l2','none'],'solver':['sag','lbfgs','newton-cg','liblinear','saga'], 'tol':[0.0001,0.00001],'max_iter':[1000,10000,100000]

GridSearchCV best estimator model comes up with parameters {'max_iter': 1000, 'penalty': 'l1', 'solver': 'liblinear', 'tol': 0.0001}

Let's build Logistic Regression model with following hyper parameters

Parameters	**Value**
solver	'liblinear'
tol	0.0001
max_iter	1000
penalty	l1

	precision	recall	f1-score	support
0	0.76	0.64	0.70	307
1	0.86	0.92	0.89	754
accuracy			0.84	1061
macro avg	0.81	0.78	0.79	1061
weighted avg	0.83	0.84	0.83	1061

	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

The above table shows the Accuracy, Precision, Recall and the F1 of the Logistic Regression model for both the train and the test data.

• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 86% for both the train and the test data for classifier 1 and 76% for training and 76% for testing data for classifier 0.

• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 92% for the train data and 88% for the test data for classifier 1 and 64% for training and 73% for testing data for classifier 0.

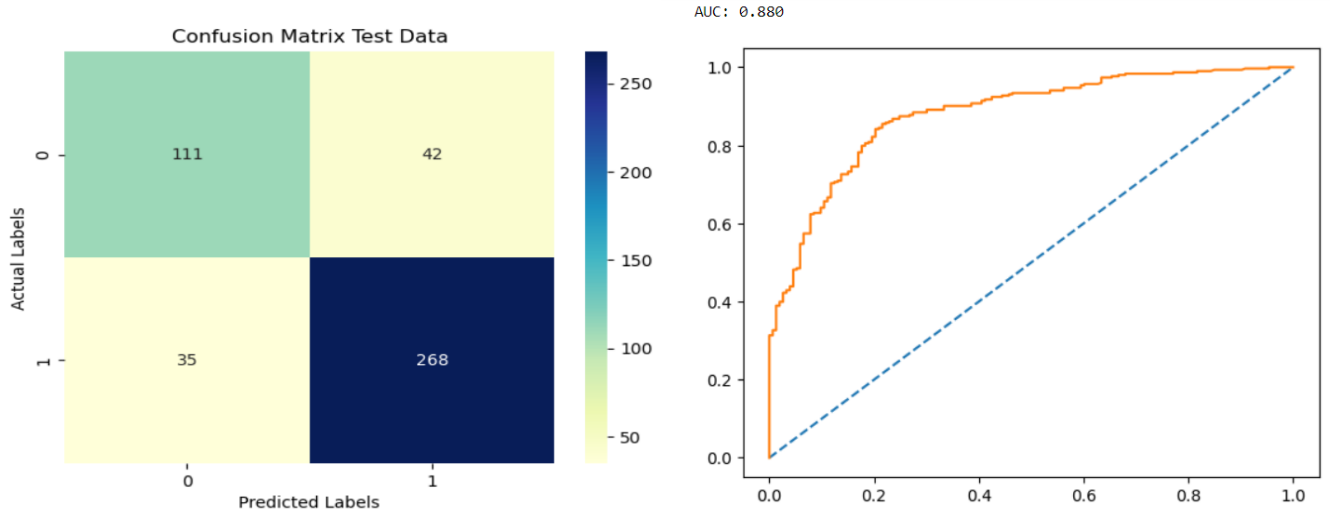
• **F1-score** : Here it is 89% for train data and 87% for test data for classifier 1 and 70% for train data and 74% for test data for classifier 0

• **Accuracy**: The train data accuracy is 84% and the test data accuracy is 83%. In this case the model accuracy is almost same for training and testing dataset so there is no over or underfitting .Accuracy has not changed in testing set after hyperparameter tuning .

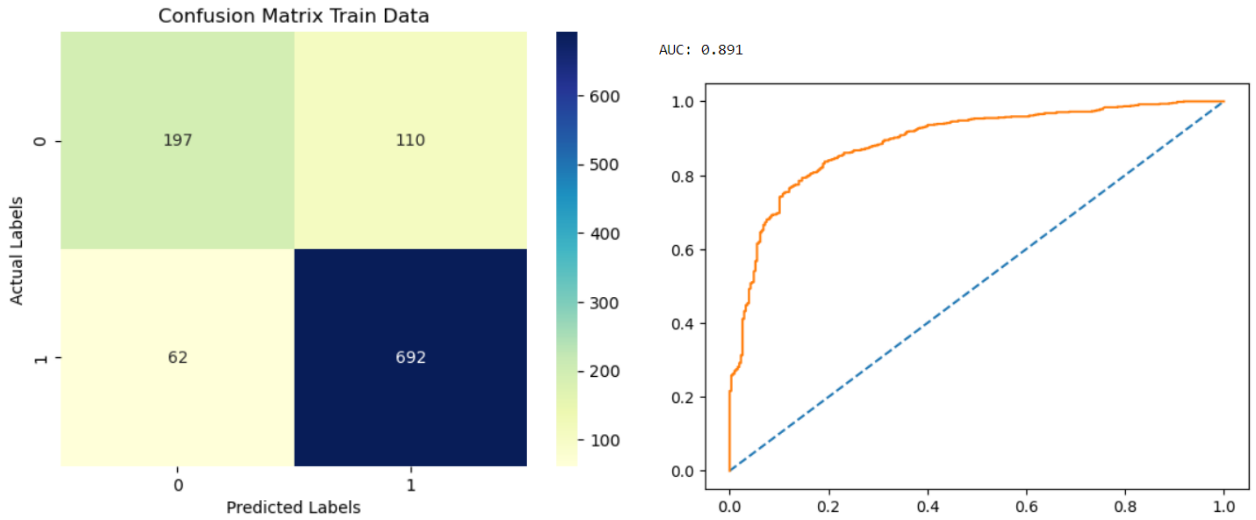
In this case the model accuracy is almost same for both training and testing dataset so there is no over or underfitting . So its a good Model

Since Class 0 is not represented less compared to Class 1 recall of Class 1 is more compared to Class 0

Feature Importance



We can from the confusion matrix above see that 111 of 153 of class 0 are precited correctly and 268 of 303 of class 1 are predicted correctly in testing data



We can from the confusion matrix above see that 197 of 307 of class 0 are precited correctly and 692 of 754 of class 1 are predicted correctly

AUC scores for testing 0.888 and training data 0.891 is very close , so there is no overfitting

KNN GridSearchCV

GridSearchCV is the process of performing hyperparameter tuning in order to determine the optimal values for a given model. The performance of a model significantly depends on the value of hyperparameters. we use GridSearchCV to automate the tuning of hyperparameters. This function helps to loop through predefined hyperparameters and fit your estimator (model) on your training set. So, in the end, we can select the best parameters from the listed hyperparameters.

The param grid that was chosen was

'weights':['uniform', 'distance'],'metric':['euclidean', 'manhattan','minkowski'], 'n_neighbors':[1,3, 5, 7, 9, 11, 13,15,17,19]

GridSearchCV best estimator model comes up with parameters { 'metric': 'manhattan', 'n_neighbors': 19, 'weights': unform}

Let's build KNN model with following hyper parameters

Parameters	**Value**
'metric'	'manhattan'
'n_neighbors	15
'weights'	`uniform`

	precision	recall	f1-score	support
0	0.75	0.69	0.72	307
1	0.88	0.91	0.89	754
accuracy			0.85	1061
macro avg	0.82	0.80	0.81	1061
weighted avg	0.84	0.85	0.84	1061

	precision	recall	f1-score	support
0	0.77	0.70	0.73	153
1	0.85	0.89	0.87	303
accuracy			0.83	456
macro avg	0.81	0.80	0.80	456
weighted avg	0.83	0.83	0.83	456

The above table shows the Accuracy, Precision, Recall and the F1 of the Logistic Regression model for both the train and the test data.

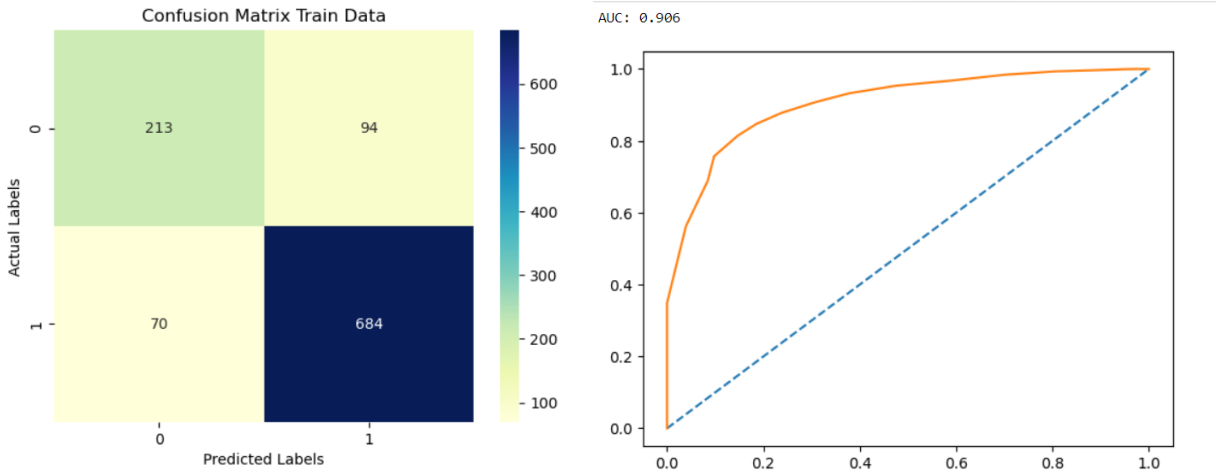
• **Precision** : it is 88% for the train and 85% for the test data for classifier 1 and 75% for training and 77% for testing data for classifier 0.

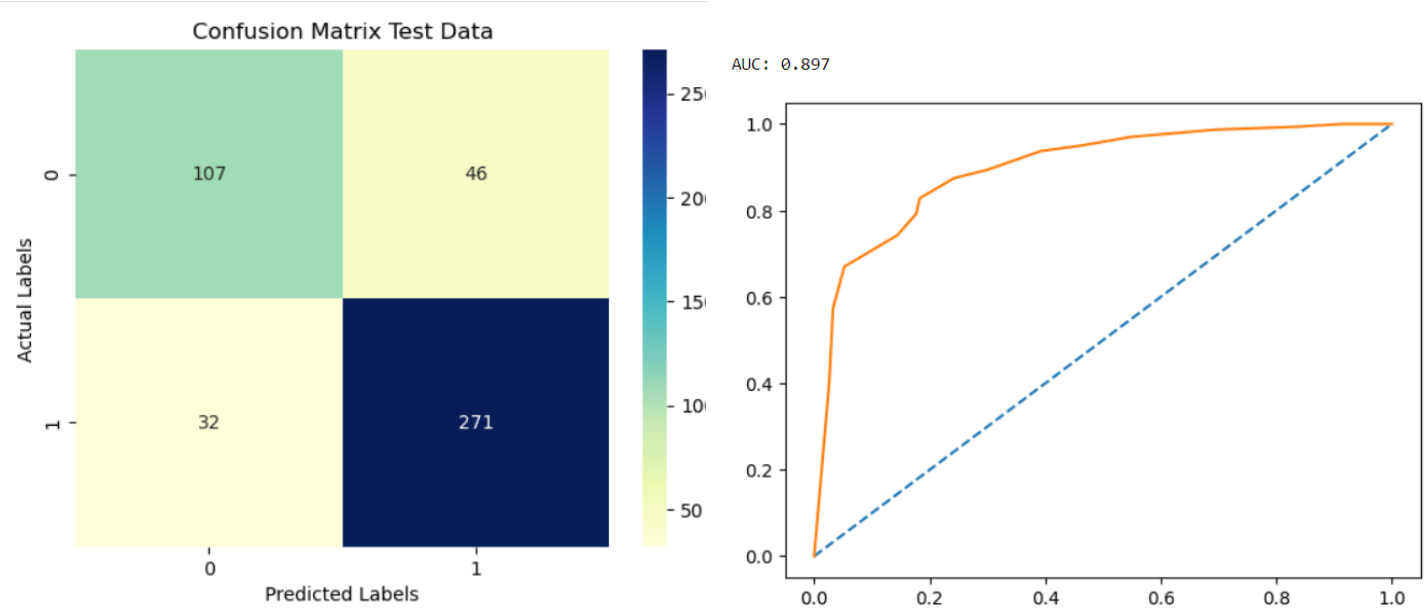
• **Recall** : it is 91% for the train data and 89 % for the test data for classifier 1 and 69% for training and 70% for testing data for classifier 0.

• **F1-score** : it is 89% for train data and 87% for test data for classifier 1 and 72% for train data and 73% for test data for classifier 0

• **Accuracy**: The train data accuracy is 85% and the test data accuracy is 83%.

In this case the model accuracy of training set is 85 % and of testing dataset is 83% so there is no over- fitting and is a suitable model





We can from the confusion matrix above see that 213 of 307 of class 0 are precited correctly and 684 of 754 of class 1 are predicted correctly

We can from the confusion matrix above see that 107 of 153 of class 0 are precited correctly and 271 of 303 of class 1 are predicted correctly in testing data

AUC scores for testing 0.897 and training data 0.906 is very close , so there is no overfitting

KNN with SMOTE

SMOTE adds synthetic data to the training sample to balance the data belonging to minority class .

As we can see 447 new observations are added to the minority class to balance the data . There are 754 records each 0 and 1 class .

	precision	recall	f1-score	support
0	0.84	0.94	0.89	754
1	0.93	0.82	0.87	754
accuracy			0.88	1508
macro avg	0.89	0.88	0.88	1508
weighted avg	0.89	0.88	0.88	1508

	precision	recall	f1-score	support
0	0.69	0.81	0.74	153
1	0.89	0.81	0.85	303
accuracy			0.81	456
macro avg	0.79	0.81	0.80	456
weighted avg	0.82	0.81	0.81	456

The above table shows the Accuracy, Precision, Recall and the F1 of the Logistic Regression model for both the train and the test data.

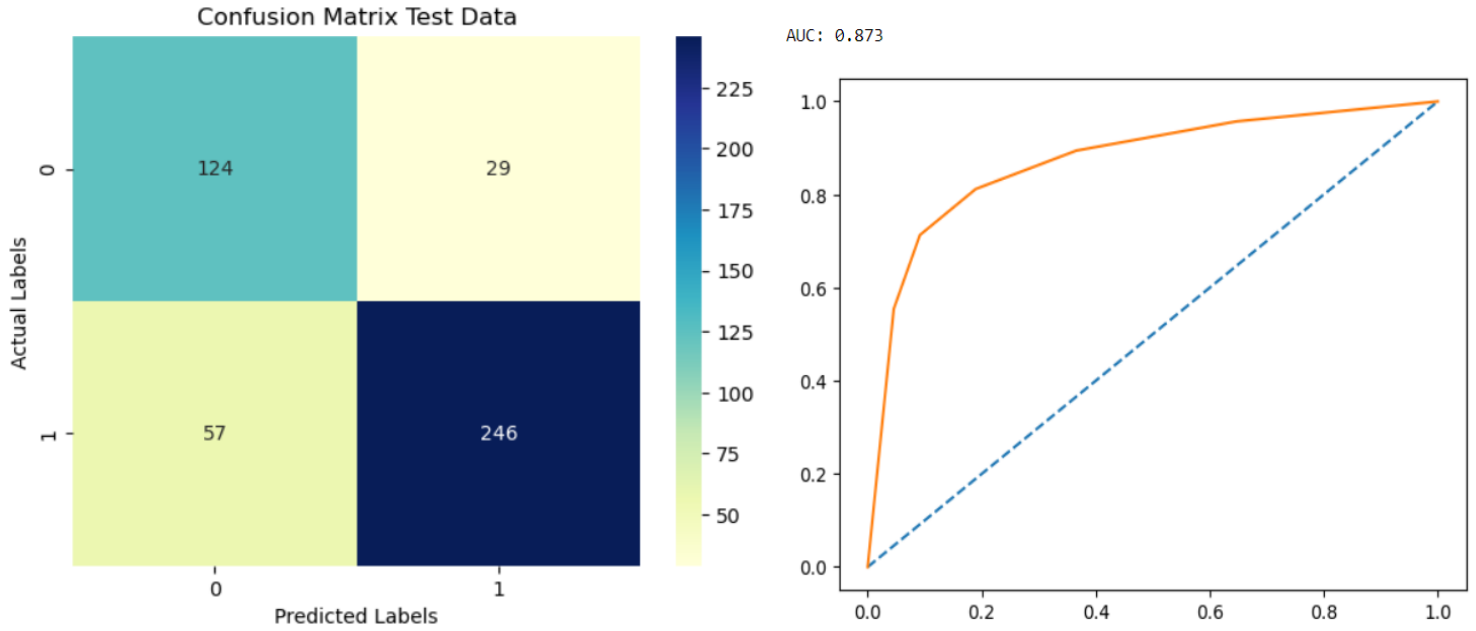
• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 93% for the train and 89% for the test data for classifier 1 and 84% for training and 69% for testing data for classifier 0.

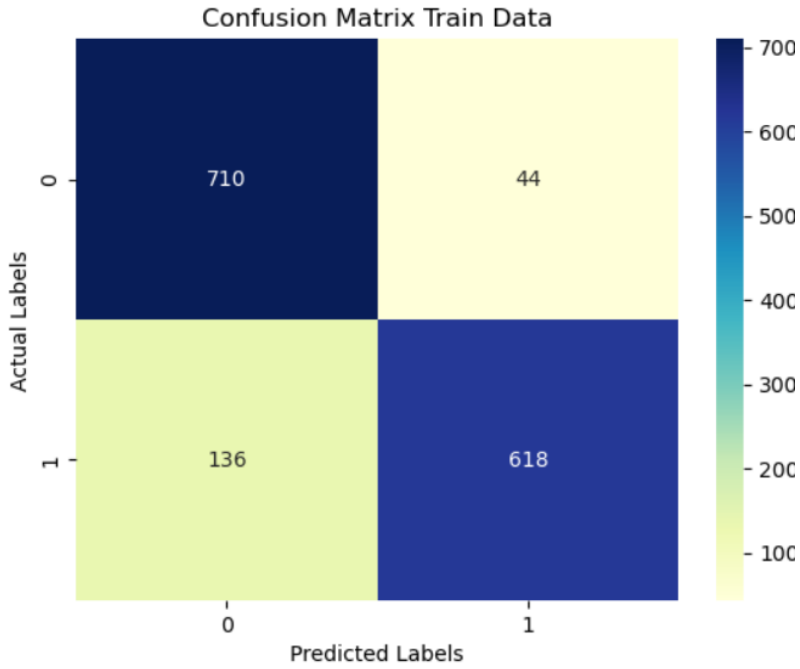
• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 82% for the train data and 81% for the test data for classifier 1 and 94% for training and 81% for testing data for classifier 0.

• **F1-score** : Here it is 87% for train data and 85% for test data for classifier 1 and 89% for train data and 74% for test data for classifier 0

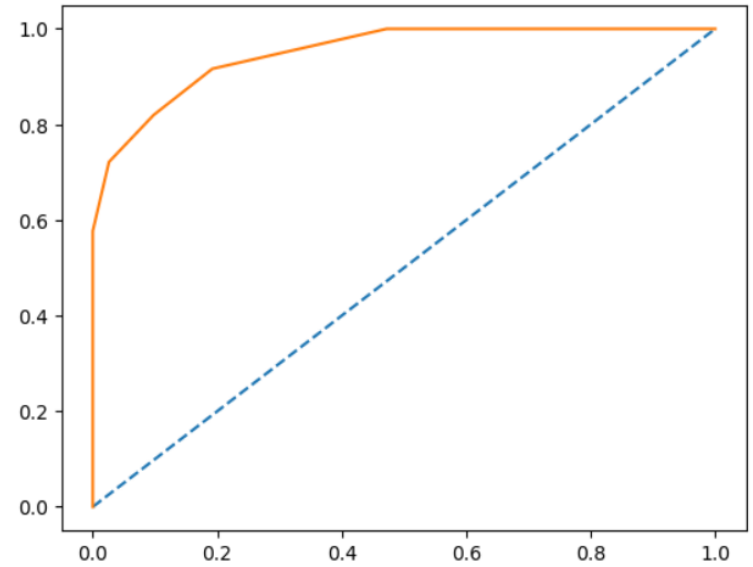
• **Accuracy**: The train data accuracy is 88% and the test data accuracy is 81%.

In this case the model accuracy of training set is 88% and of testing dataset is 81% so there is no over- fitting and is a suitable model .


Confusion Matrix Test Data
124
29
57
246
0
1
Actual Labels
Predicted Labels
225
200
175
150
125
100
75
50
AUC: 0.873
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Confusion Matrix Train Data
710
44
136
618
0
1
Actual Labels
0
1
Predicted Labels
700
600
500
400
300
200
100

```
AUC: 0.950
```



We can from the confusion matrix above see that 710 of 754 of class 0 are precited correctly and 618 of 754 of class 1 are predicted correctly in training data

We can from the confusion matrix above see that 124 of 153 of class 0 are precited correctly and 246 of 303 of class 1 are predicted correctly in testing data

AUC scores for testing 0.873 and training data 0.95, so there is no overfitting.

Naive Bayes with SMOTE

SMOTE adds synthetic data to the training sample to balance the data belonging to minority class .

As we can see 447 new observations are added to the minority class to balance the data . There are 754 records each 0 and 1 class .

	precision	recall	f1-score	support
0	0.82	0.83	0.82	754
1	0.82	0.81	0.82	754
accuracy			0.82	1508
macro avg	0.82	0.82	0.82	1508
weighted avg	0.82	0.82	0.82	1508

	precision	recall	f1-score	support
0	0.67	0.81	0.73	153
1	0.89	0.80	0.84	303
accuracy			0.80	456
macro avg	0.78	0.80	0.79	456
weighted avg	0.82	0.80	0.80	456

The above table shows the Accuracy, Precision, Recall and the F1 of the Logistic Regression model for both the train and the test data.

• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 82% for the train and 89% for the test data for classifier 1 and 82% for training and 67% for testing data for classifier 0.

• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 81% for the train data and 80% for the test data for classifier 1 and 83% for training and 81% for testing data for classifier 0.

• **F1-score** : Here it is 82% for train data and 84% for test data for classifier 1 and 82% for train data and 73% for test data for classifier 0

• **Accuracy**: The train data accuracy is 82% and the test data accuracy is 80%.

When a model performs very well in the train data and but doesn't perform well in the testing set performs ,it's called overfitting and the inverse is called the underfitting. In this case the model accuracy is slightly different for both training and testing dataset so there is no over or underfitting .

Bagging

Bagging is an ensemble algorithm that fits multiple models on different subsets of a training dataset, then combines the predictions from all models.

Random forest is an extension of bagging that also randomly selects subsets of features used in each data sample. Both bagging and random forests have proven effective on a wide range of different predictive modelling problems.

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Bagging with Decision Tree classifier

```
0.8092105263157895
[[100  53]
 [ 34 269]]
              precision    recall  f1-score   support

           0       0.75      0.65      0.70       153
           1       0.84      0.89      0.86       303

    accuracy                           0.81       456
   macro avg       0.79      0.77      0.78       456
weighted avg       0.81      0.81      0.81       456

0.9905749293119699
[[301   6]
 [  4 750]]
              precision    recall  f1-score   support

           0       0.99      0.98      0.98       307
           1       0.99      0.99      0.99       754

    accuracy                           0.99      1061
   macro avg       0.99      0.99      0.99      1061
weighted avg       0.99      0.99      0.99      1061
```

The above table shows the Accuracy, Precision, Recall and the F1 of the Bagging with Decision Tree classifier model for both the train and the test data.

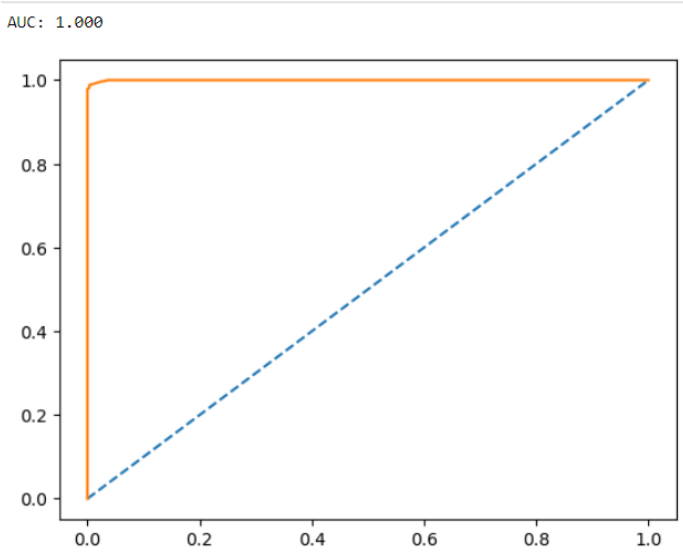
• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 99% for the train and 84% for the test data for classifier 1 and 99% for training and 75% for testing data for classifier 0.

• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 99% for the train data and 89% for the test data for classifier 1 and 98% for training and 65% for testing data for classifier 0.

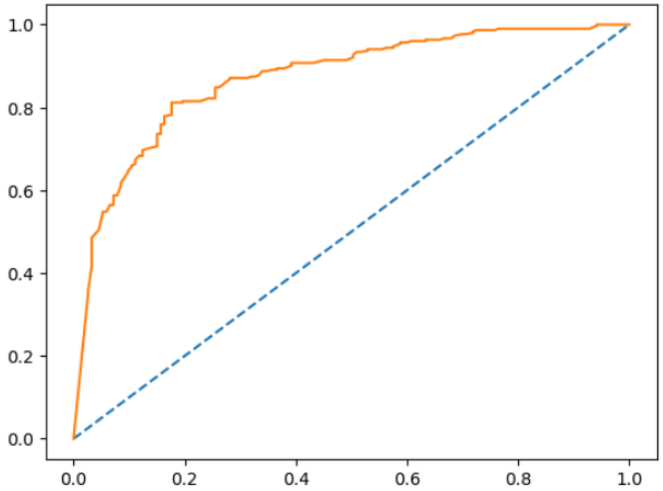
• **F1-score** : Here it is 99% for train data and 86% for test data for classifier 1 and 98% for train data and 70% for test data for classifier 0

• **Accuracy**: The train data accuracy is 99% and the test data accuracy is 81%.

When a model performs very well in the train data and but doesn't perform well in the testing set performs ,it's called overfitting and the inverse is called the underfitting. In this case the model accuracy for training is very high compared to testing dataset so there is overfitting.



```
AUC: 0.870
```



Bagging with Random Forest classifier

```
0.8333333333333334
[[101  52]
 [ 24 279]]
              precision    recall  f1-score   support

           0       0.81      0.66      0.73       153
           1       0.84      0.92      0.88       303

    accuracy                           0.83       456
   macro avg       0.83      0.79      0.80       456
weighted avg       0.83      0.83      0.83       456

0.9622997172478793
[[274  33]
 [  7 747]]
              precision    recall  f1-score   support

           0       0.98      0.89      0.93       307
           1       0.96      0.99      0.97       754

    accuracy                           0.96      1061
   macro avg       0.97      0.94      0.95      1061
weighted avg       0.96      0.96      0.96      1061
```

The above table shows the Accuracy, Precision, Recall and the F1 of the Bagging with Decision Tree classifier model for both the train and the test data.

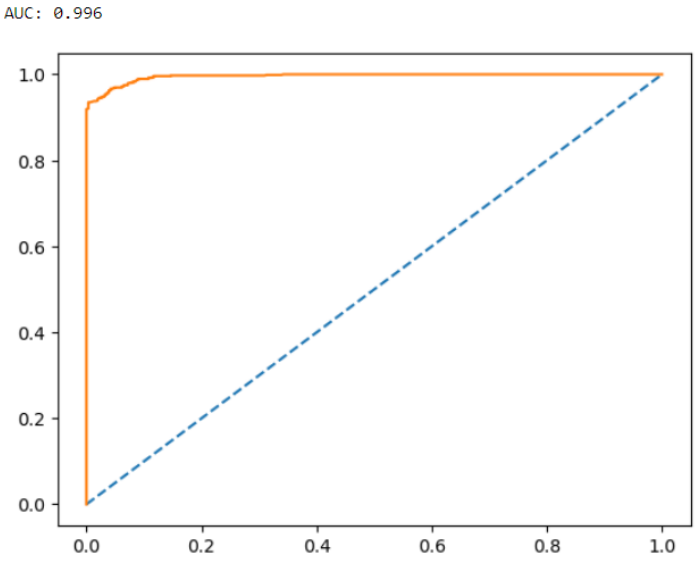
• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 96% for the train and 84% for the test data for classifier 1 and 98% for training and 81% for testing data for classifier 0.

• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 99% for the train data and 92% for the test data for classifier 1 and 89% for training and 66% for testing data for classifier 0.

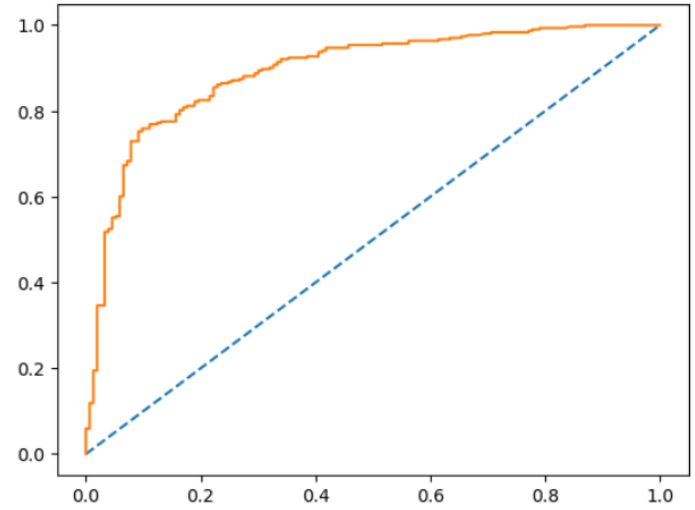
• **F1-score** : Here it is 97% for train data and 88% for test data for classifier 1 and 93% for train data and 73% for test data for classifier 0

• **Accuracy**: The train data accuracy is 96% and the test data accuracy is 83%.

When a model performs very well in the train data and but doesn't perform well in the testing set performs ,it's called overfitting and the inverse is called the underfitting. In this case the model accuracy for training is very high compared to testing dataset so there is overfitting.



```
AUC: 0.892
```



Ada Boosting

```
0.8482563619227145
[[212  95]
 [ 66 688]]
              precision    recall  f1-score   support

           0       0.76      0.69      0.72       307
           1       0.88      0.91      0.90       754

    accuracy                           0.85      1061
   macro avg       0.82      0.80      0.81      1061
weighted avg       0.85      0.85      0.85      1061
```

```
0.8245614035087719
[[105  48]
 [ 32 271]]
              precision    recall  f1-score   support

           0       0.77      0.69      0.72       153
           1       0.85      0.89      0.87       303

    accuracy                           0.82       456
   macro avg       0.81      0.79      0.80       456
weighted avg       0.82      0.82      0.82       456
```

The above table shows the Accuracy, Precision, Recall and the F1 of the Bagging with Decision Tree classifier model for both the train and the test data.

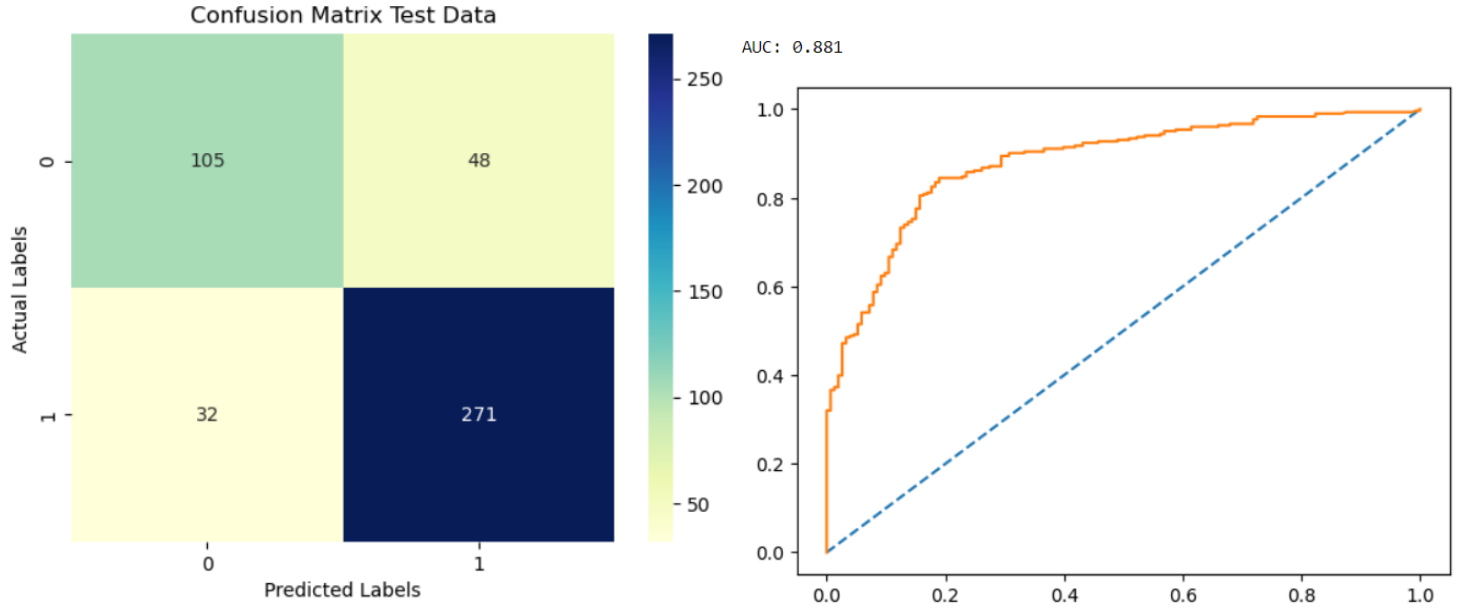
• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 88% for the train and 85% for the test data for classifier 1 and 76% for training and 77% for testing data for classifier 0.

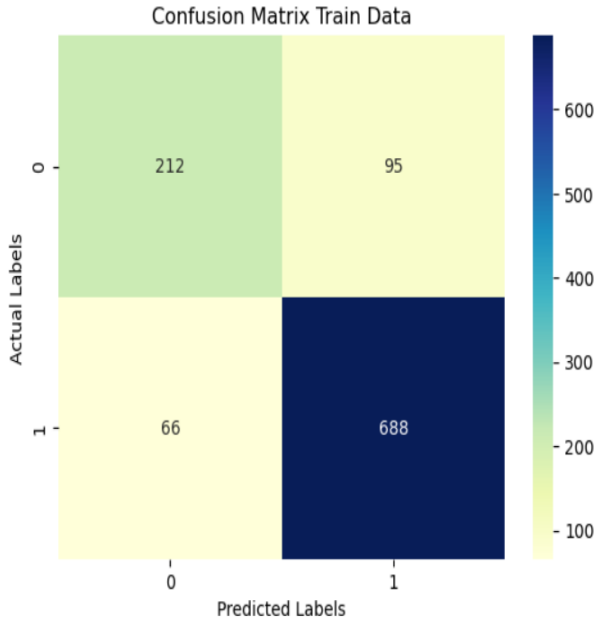
• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 91% for the train data and 89% for the test data for classifier 1 and 69% for training and 69% for testing data for classifier 0.

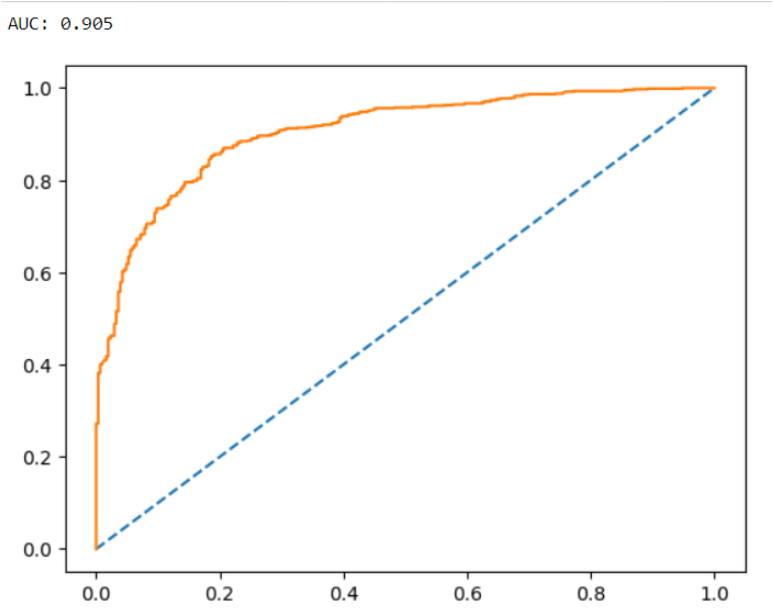
• **F1-score** : Here it is 90% for train data and 87% for test data for classifier 1 and 72% for train data and 72% for test data for classifier 0

• **Accuracy**: The train data accuracy is 85% and the test data accuracy is 82%.

When a model performs very well in the train data and but doesn't perform well in the testing set performs ,it's called overfitting and the inverse is called the underfitting. In this case the model accuracy for training similar to testing dataset so there is no overfitting.







Gradient Boosting

0.8821800103990235

	precision	recall	f1-score	support
0	0.82	0.77	0.79	307
1	0.91	0.93	0.92	754
accuracy			0.88	1061
macro avg	0.86	0.85	0.85	1061
weighted avg	0.88	0.88	0.88	1061

	precision	recall	f1-score	support
0	0.81	0.66	0.73	153
1	0.84	0.92	0.88	303
accuracy			0.83	456
macro avg	0.83	0.79	0.80	456
weighted avg	0.83	0.83	0.83	456

The above table shows the Accuracy, Precision, Recall and the F1 of the Bagging with Decision Tree classifier model for both the train and the test data.

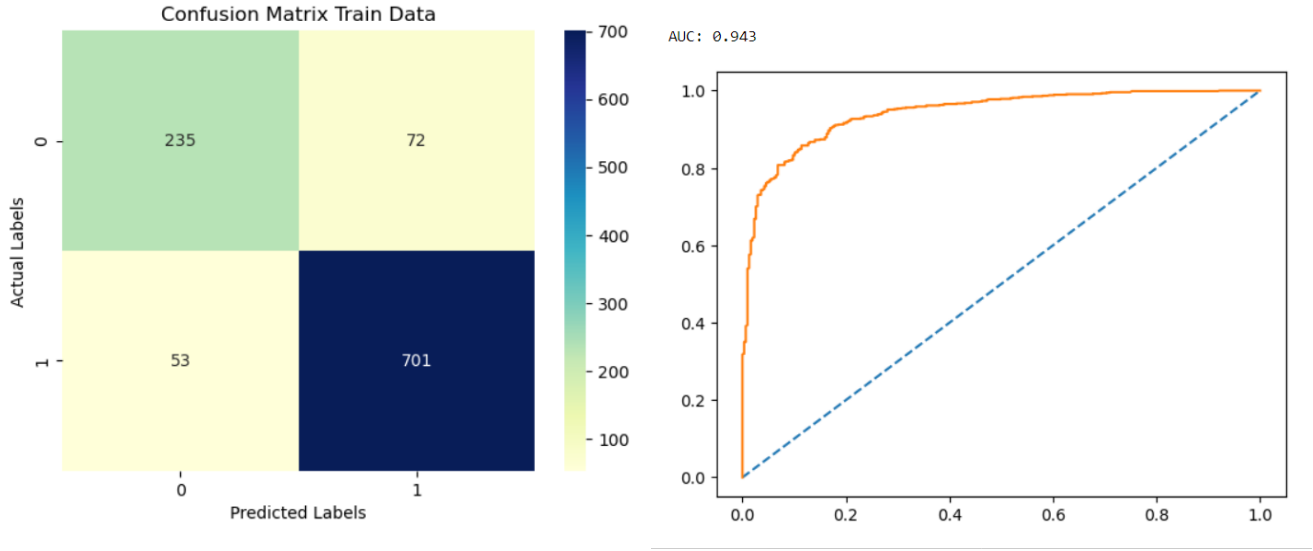
• **Precision** : It is the number of true positive results divided by the number of positive results predicted by the classifier. Here it is 91% for the train and 84% for the test data for classifier 1 and 82% for training and 81% for testing data for classifier 0.

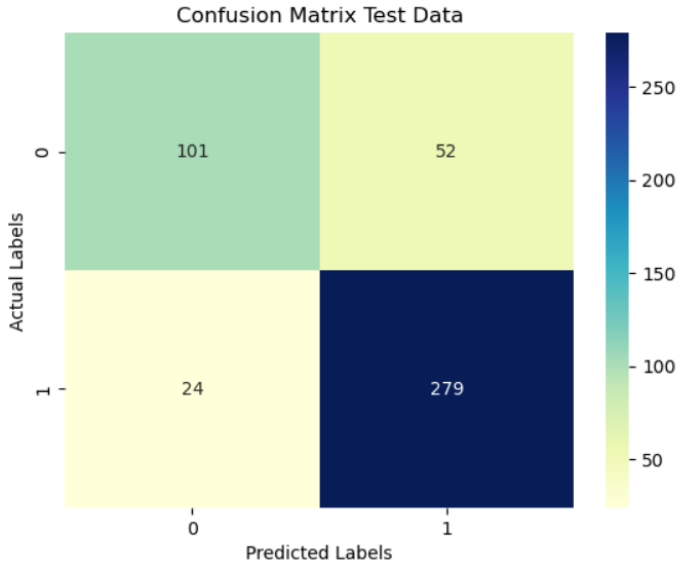
• **Recall** : It is the number of true positive results divided by number of all samples that should have been identified as positive. Here it is 93% for the train data and 92% for the test data for classifier 1 and 76% for training and 66% for testing data for classifier 0.

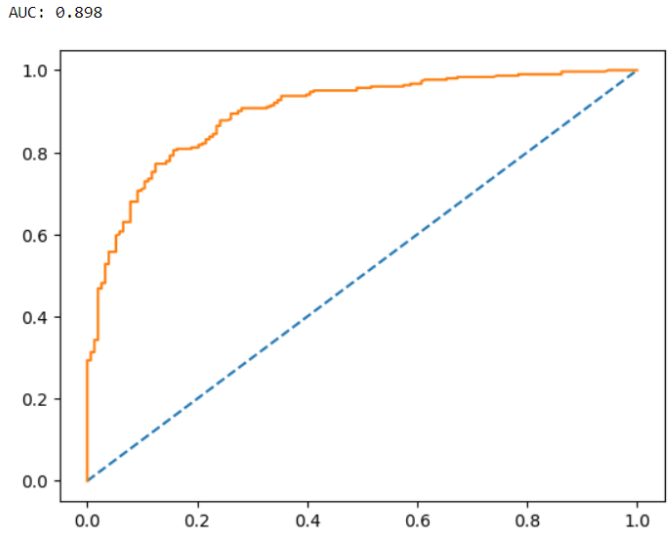
• **F1-score** : Here it is 92% for train data and 88% for test data for classifier 1 and 79% for train data and 73% for test data for classifier 0

• **Accuracy**: The train data accuracy is 88% and the test data accuracy is 83%.

When a model performs very well in the train data and but doesn't perform well in the testing set performs ,it's called overfitting and the inverse is called the underfitting. In this case the model accuracy for training sighlty different from testing dataset so there is no overfitting or underfitting







Best Model is KNN model with accuracy of 85 % and AUC: 0.898

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.

Key Insights from the Model Building and EDA on the above dataset is as follows .

- The Assessment of the Labour leader and Conservative leader Variables 'Blair' and 'Hague' are the most importance features in the predicting the outcome of the exit polls .A voter who rated Mr Blair as 5 is more likely to vote for Labour, whereas the one who rated Hague as 5 and Mr Blair as 1 is more likely to vote for Conservative
- The second important variable is the European integration of UK .Respondents with high Eurosceptic scores are likely to vote for Conservative party whereas respondents with low Eurosceptic scores are likely to vote for Labour party,
- Taking age group criteria, we can see that older group 65-80 prefer conservative party and age group 35-50 prefer Labour Party
- Conservative party supports are more likely to be aware of the political situation than Labour party voters .Gender of the voter is not a significant factor in predicting outcome of the polls .
- Labour party is most preferred party and most likely to win the elections based on the exit polls

Recommendations

- Of all the machine models' KNN Model seems to performing slightly better than other models because AUC score of knn Regression Model is slightly higher compared to LDA and Logistuc
- All Bagging Models seems to be performing poorly on testing dataset but doing exceedingly well on training dataset so there is problem of overfitting and is not recommended

- Conservative party is not very popular in general and also with younger generation and hence they need to target the younger generation in order to improve their chance of the winning the elections.

Problem2

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

(Hint: use .words(), .raw(), .sent() for extracting counts)

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

Number of Characters in Speech of President Franklin D. Roosevelt in 1941 are 7571
Number of Characters in Speech of President John F. Kennedy in 1961 are 7618
Number of Characters in Speech of President Richard Nixon in 1973 are 9991

Number of words in Speech of President Franklin D. Roosevelt in 1941 are 1536
Number of words in Speech of President John F. Kennedy in 1961 are 1546
Number of words in Speech of President Richard Nixon in 1973 are 2028

Number of words in Speech of President Franklin D. Roosevelt in 1941 are 68
Number of words in Speech of President John F. Kennedy in 1961 are 52
Number of words in Speech of President Richard Nixon in 1973 are 69

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

3

Number of words in Speech of President Franklin D. Roosevelt before removal of stop words is 1536
Number of words in Speech of President John F. Kennedy before removal of stop words is 1546
Number of words in Speech of President Richard Nixon before removal of stop words is 2028
Number of words in Speech of President Franklin D. Roosevelt after removal of stop words is : 632
Number of words in Speech of President John F. Kennedy after removal of stop words is : 697
Number of words in Speech of President Richard Nixon after removal of stop words is : 836

Sample sentence
The world is very different now. For man holds in his mortal hands the power to abolish all forms of h
uman poverty and all forms of human life

Filtered sentence after removal of stopwords

['world', 'different', 'man', 'holds', 'mortal', 'hands', 'power', 'abolish', 'forms', 'human', 'poverty', 'forms', '
human', 'life']

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)

Words that occured most in speech of President Franklin D. Roosevelt after removal of stopwords
is ‘nation' which occurred 12 times followed by 'know' 10 times and 'spirit' ,’life’ and ‘democracy’ occ
urring 9 times

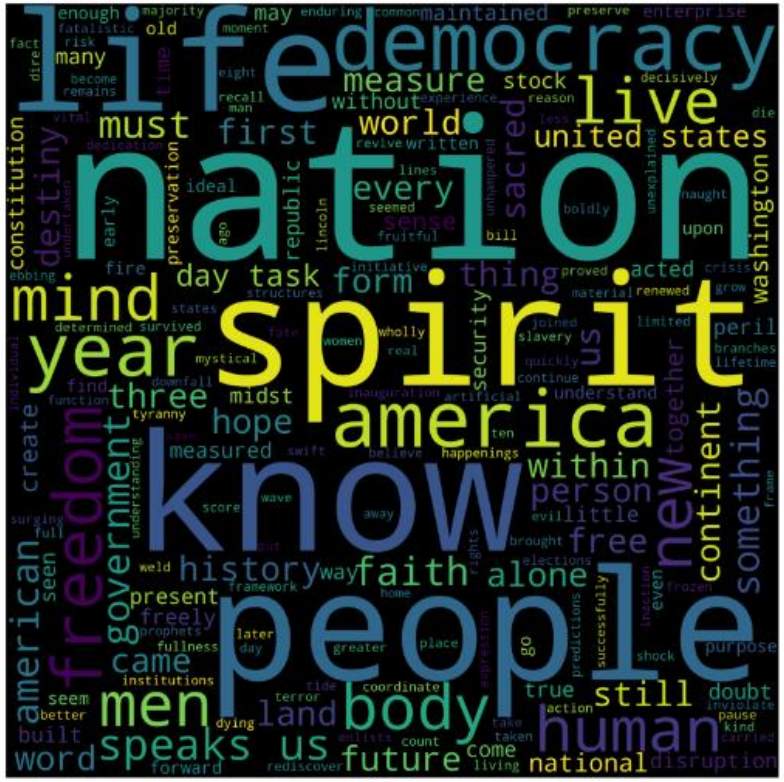
3

Words that occured most in speech of President John F. Kennedy after removal of stopwords
is 'let' which occurred 16 times followed by 'us' 12 times and 'world' and ‘sides’ 8 times

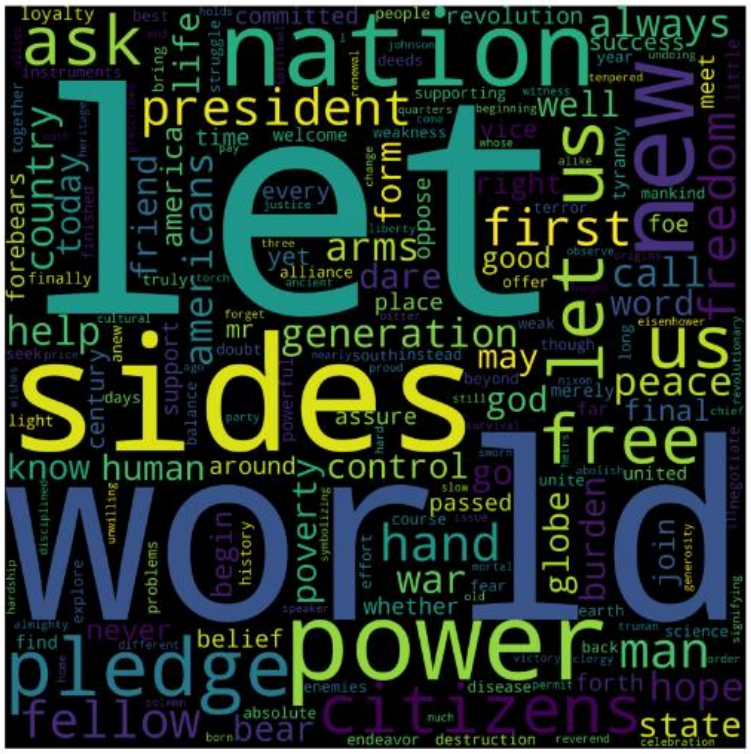
Words that occured most in speech of President Richard Nixon , after removal of stopwords
Is 'us' which occurred 26 times followed by 'let' that occurred 22 times and 'america' 21 times

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

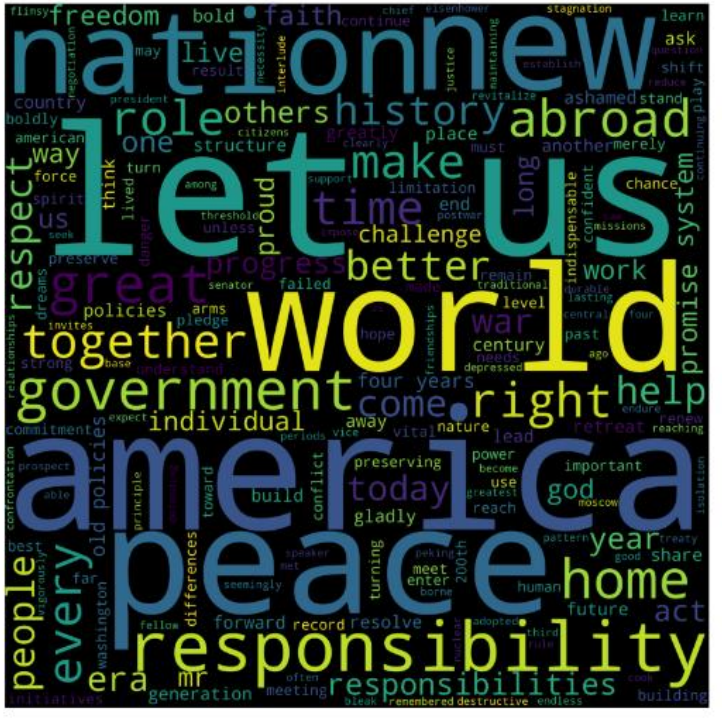
Word Cloud for President Roosevelt (after cleaning)!!



Word Cloud for President Kennedy (after cleaning)!!



Word Cloud for President Nixon (after cleaning)!!



Problem 2: