MACHINE LEARNING PROJECT REPORT

By

Pankul Dhamanda

Contents		No.
Problem 1 You are hired by one of the leading news channels CNBE who wants to analyze recent elections. This survey was conducted on 1525 voters with 9 variables. You have to build a model, to predict which party a voter will vote for on the basis of the given information, to create an exit poll that will help in predicting overall win and seats covered by a particular party.		6
1.1	Read the dataset. Do the descriptive statistics and do the null value condition check. Write an inference on it.	6
1.2	Perform Univariate and Bivariate Analysis. Do exploratory data analysis. Check for Outliers.	9
1.3	Encode the data (having string values) for Modelling. Is Scaling necessary here or not? Data Split: Split the data into train and test (70:30).	14
1.4	Apply Logistic Regression and LDA (linear discriminant analysis).	16
1.5	Apply KNN Model and Naïve Bayes Model. Interpret the results.	19
1.6	Model Tuning, Bagging (Random Forest should be applied for Bagging), and boosting.	22
1.7	Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model. Final Model: Compare the models and write inference which model is best/optimized.	34
1.8	Based on these predictions, what are the insights?	55
Problem 2 In this particular project, we are going to work on the inaugural corpora from the nltk in Python. We will be looking at the following speeches of the Presidents of the United States of America: 1. President Franklin D. Roosevelt in 1941 2. President John F. Kennedy in 1961 3. President Richard Nixon in 1973 (Hint: use. words(), .raw(), .sent() for extracting counts)		57
2.1	Find the number of characters, words, and sentences for the mentioned documents.	57
2.2	Remove all the stop words from all three speeches.	58
2.3	Which word occurs the most number of times in his inaugural address for each president? Mention the top three words. (after removing the stop words)	59
2.4	Plot the word cloud of each of the speeches of the variable. (after removing the stop words)	60

List of Figures		No.
1.	Head of the dataset	6
2.	Processed dataset	6
3.	Null values	7
4.	Information of the dataset	7
5.	Description of the dataset	7
6.	Skewness of Data	8
7.	Histogram and boxplot	9
8.	Count plot for categorical variables	10
9.	Count plot showing bivariate analysis of different components	10
10.	Strip plot showing bivariate analysis between Votes and Europe	11
11.	Strip plot showing bivariate analysis between Votes and age	11
12.	Pair plot showing bivariate analysis	12
13.	Heatmap showing correlation between different components	13
14.	Encoded Dataset	14
15.	Encoded information of dataset	14
16.	Scaled dataset	15
17.	LR-Classification report- Train data	16
18.	LR-Classification report- Test data	16
19.	LDA-Classification report- Train data	17
20.	LDA-Classification report- Test data	17
21.	KNN- Classification report- Train data	19
22.	KNN- Classification report- Test data	19
23.	NB- Classification report- Train data	20
24.	NB- Classification report- Test data	20
25.	RF Feature Importance	29
26.	Confusion matrix and classification report of LR Model	34
27.	Roc-Auc curve LR Model	35
28.	Confusion matrix, Classification report and AUC-ROC curve-LDA Model	36
29.	Confusion matrix, Classification report and AUC-ROC curve-KNN Model	37
30.	Confusion matrix, Classification report and AUC-ROC curve of Naïve Bayes Model	38
31.	Confusion matrix, Classification report and AUC-ROC curve of LR Model (Tuned)	39
32.	Confusion matrix, Classification report and AUC-ROC curve of LDA(Tuned)	40
33.	Confusion matrix, Classification report and AUC-ROC curve of KNN Model (Tuned)	41
34.	Confusion matrix, Classification report and AUC-ROC curve of Naïve Bayes Model (Tuned)	42
35.	Confusion matrix, Classification report and AUC-ROC curve of Bagging Model	43
36.	Confusion matrix, Classification report and AUC-ROC curve of Bagging Model (Tuned)	44
37.	Confusion matrix, Classification report and AUC-ROC curve of Random Forest	45
38.	Confusion matrix, Classification report and AUC-ROC curve of Random Forest (Tuned)	46
39.	Confusion matrix, Classification report and AUC-ROC curve of Ada Boost Model	47
40.	Confusion matrix, Classification report and AUC-ROC curve of Ada Boost Model (Tuned)	48
41.	Confusion matrix, Classification report and AUC-ROC curve of Gradient Boost Model	49
42.	Confusion matrix, Classification report and AUC-ROC curve of Gradient Boost Model (Tuned)	50
43.	Comparison of AUC ROC curve on train data for all the Models	53
44.	Comparison of AUC ROC curve on test data for all the Models	54

45.	Dataset after removal of stop words	58
46.	Word cloud of Roosevelt's speech	60
47.	Word cloud of Kennedy’s speech	61
48.	Word cloud of Nixon’s speech	62

List of Tables		No.
1.	Shape of Test Train split data	15
2.	Comparison of Simple v/s tuned-LR Model	22
3.	Comparison of Simple v/s tuned-LDA Model	23
4.	Comparison of Simple v/s tuned-KNN Model	24
5.	Comparison of Simple v/s tuned- Naïve Bayes Model	25
6.	Comparison of Simple v/s tuned Bagging Model	27
7.	Comparison of Simple v/s tuned-Random Forest Model	29
8.	Comparison of Simple v/s tuned-Ada Boost Model	31
9.	Comparison of Simple v/s tuned-Gradient Boost Model	33
10.	Comparison of train data for all the Models	51
11.	Comparison of test data for all the Models	52

Data Dictionary for Election Data Survey	
Vote	Party choice: conservative or Labour
Age	In years
Economic Cond national	Assessment of current national economic conditions, 1 to 5.
Economic Cond household	Assessment of current household economic conditions, 1 to 5.
Blair	Assessment of the Labour leader, 1 to 5.
Hague	Assessment of the conservative leader, 1 to 5.
Europe	An 11-point scale that measures respondents' attitudes toward European integration. High scores represent Euro skeptical sentiment.
Political knowledge	Knowledge of parties' positions on European integration, 0 to 3.
gender	Male or Female

Problem 1

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

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

In this case study, we will examine how we can approach decisions related to one of the leading news channels CNBE who wants to analyze recent elections. After importing the dataset, we check the shape of the data, the types of variables, missing values and duplicate values in the dataset.

First five rows of the dataset are shown below

	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

Fig.1 Head of the dataset

<u>Observation</u>

- We have dropped the 'unnamed' column as it is not useful for our study.
- The Dataset contains 8 duplicate values, which needs to be dropped.
- After **dropping 'unnamed' column and 8 duplicate values**, below output is obtained.

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

Fig.2 Processed dataset

- There are no null values in the dataset.

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

Fig.3 Null values

- The info function shows that dataset comprises of **1517 rows** and **9 columns** with **7 integer** and **2 object** variables.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1517 entries, 0 to 1524
Data columns (total 9 columns):
 #   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)
memory usage: 118.5+ KB
```

Fig.4 Information of the dataset

- The **describe method** provides statistical information of the numerical values in the dataset like the mean, median, max. and min. values etc.
- The Highest age is 93.

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge
count	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000
mean	54.182295	3.245902	3.140328	3.334426	2.746885	6.728525	1.542295
std	15.711209	0.880969	0.929951	1.174824	1.230703	3.297538	1.083315
min	24.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000
25%	41.000000	3.000000	3.000000	2.000000	2.000000	4.000000	0.000000
50%	53.000000	3.000000	3.000000	4.000000	2.000000	6.000000	2.000000
75%	67.000000	4.000000	4.000000	4.000000	4.000000	10.000000	2.000000
max	93.000000	5.000000	5.000000	5.000000	5.000000	11.000000	3.000000

Fig.5 Description of the dataset

Skewness of Dataset

```
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
dtype: float64
```

Fig.6 Skewness of Data

- There isn't much skewness in the data as all the values range between -0.5 and 0.5.
- The value of Blair is slightly higher than 0.5.
- Overall, the data is symmetric.

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

<u>**Univariate Analysis (Numerical Variables)**</u>

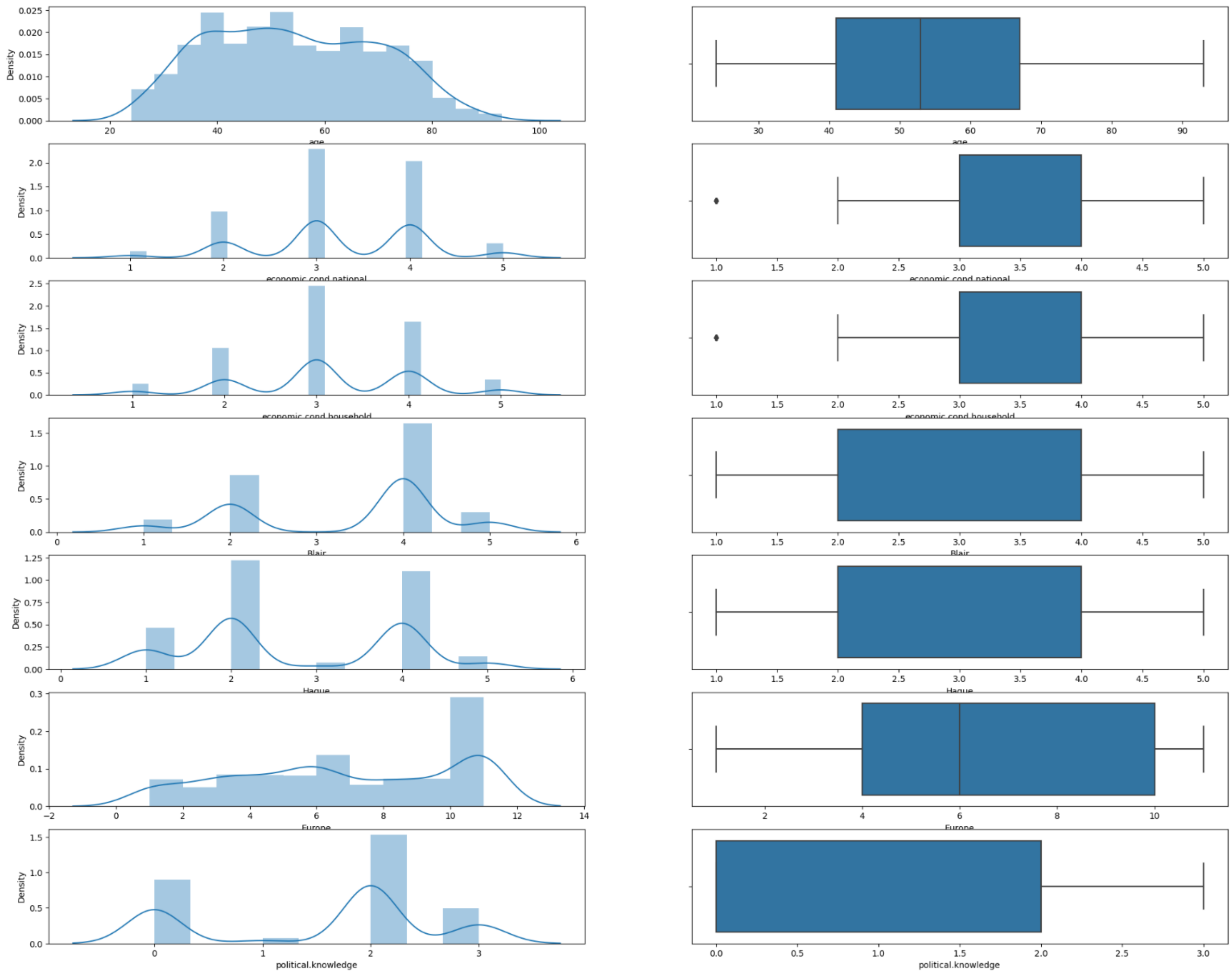


Fig.7 Histogram and boxplot

<u>**Observation**</u>

- Most of the people are aged between 40 and 70.
- The average values of economic.cond.national and economic.cond.household are 3.2459 and 3.14 respectively.
- Outliers are present in economic.cond.national and economic.cond.household.
- Variable 4 has the highest value in Blair which is 833.
- Out of 1517 people 454 do not have any knowledge of parties' positions in European integration.
- **There are no outliers present in the continuous variable 'age'. The remaining variables are categorical in nature.**

Univariate Analysis (Categorical Variables)

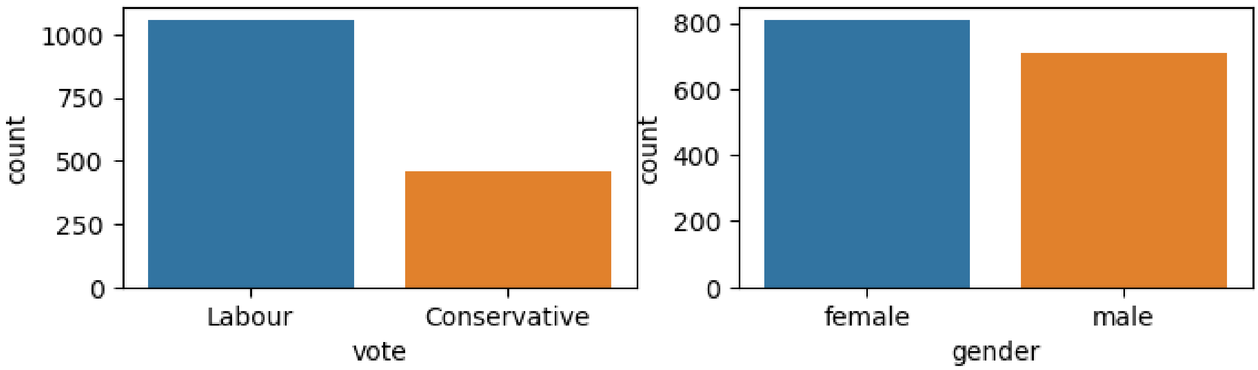


Fig.8 Count plot for categorical variables

Observation

- Here, Labour Party votes (1057) accounts for more than double the votes of Conservative Party votes (460).
- The number of female voters is 812 which is slightly higher than the 713 number of male voters.

Bivariate Analysis

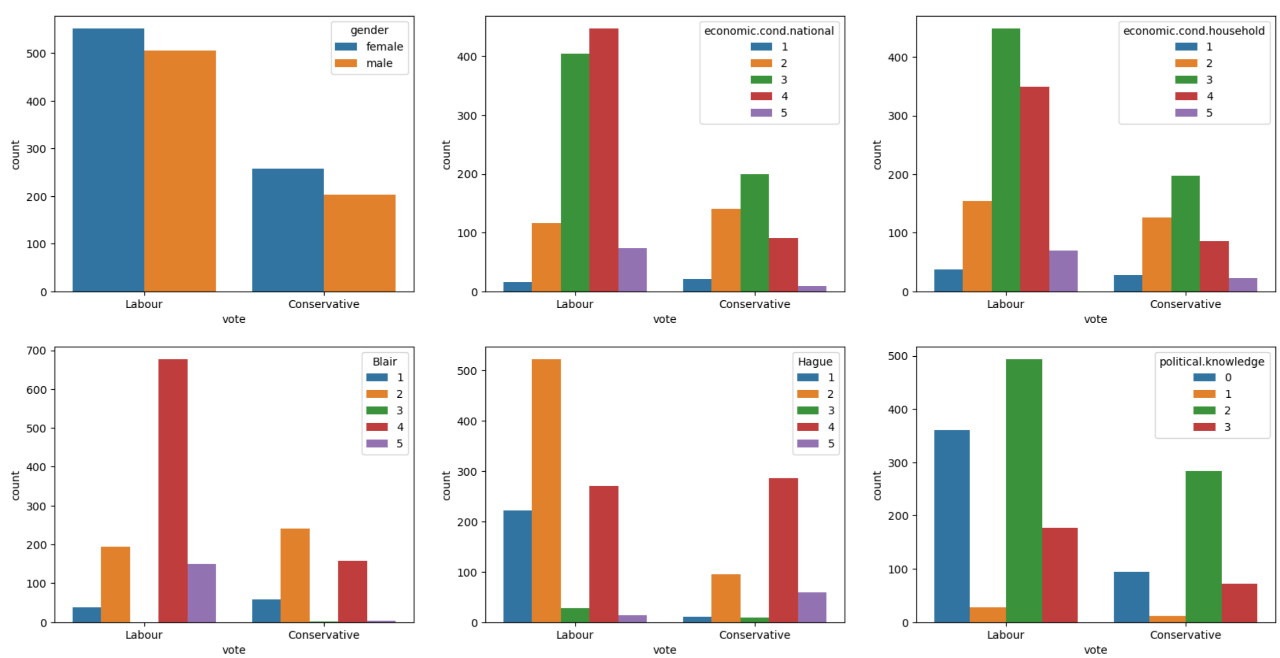


Fig.9 Count plot showing bivariate analysis of different components

Observation

- It can be clearly seen that overall, the Labour party has got more votes than conservative party.
- The score of 3, 4 and 5 have more votes in the Labour party
- The score of 1 and 2 have more votes in the conservative party.
- A significant percentage of people who gave a bad score to the conservative leader still choose to vote for 'Hague'.

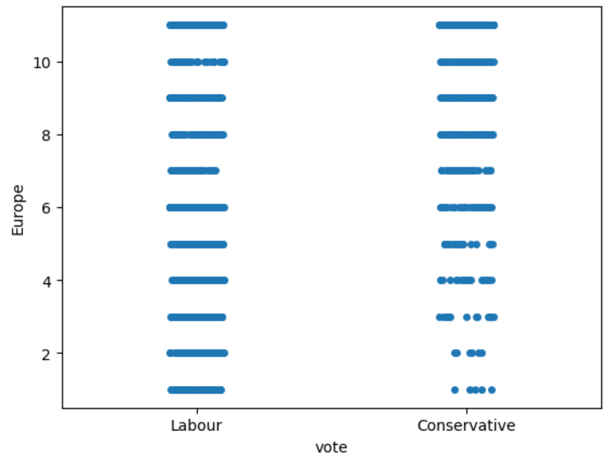


Fig.10 Strip plot showing bivariate analysis between Votes and Europe

- We can infer that lower the 'Eurosceptic' sentiment, higher the votes for Labour party.

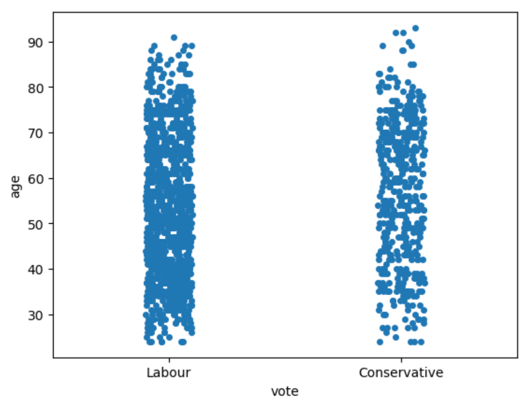


Fig.11 Strip plot showing bivariate analysis between Votes and age

- Labour party has got more votes than conservative party across all age groups.

Pair plot (Bivariate Analysis)

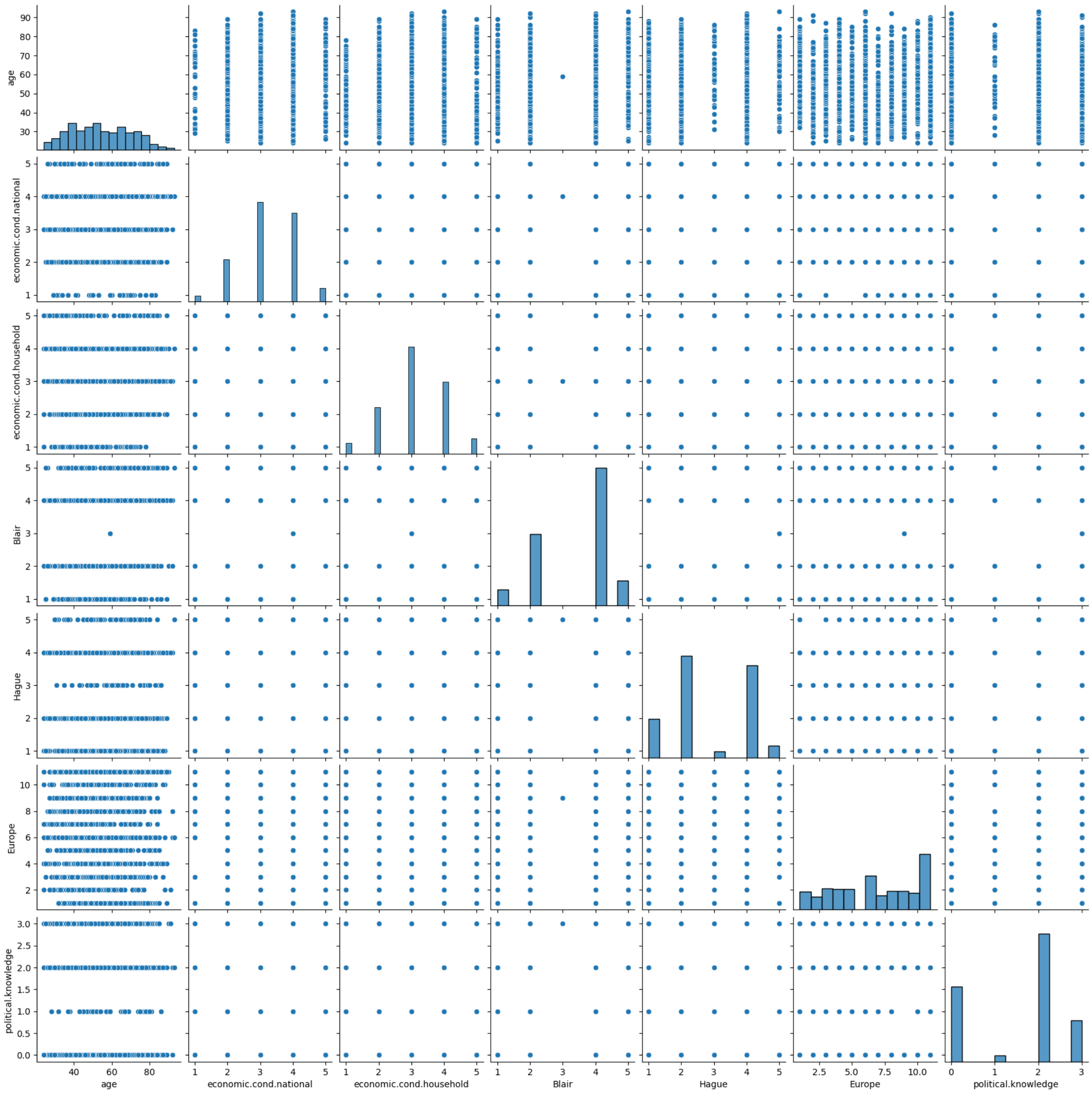


Fig.12 Pair plot showing bivariate analysis

Observation

- It can be observed that Blair, Europe and political knowledge are left skewed.
- Most of the variables are normally distributed.
- Also, it can be observed from the scatterplots that there is almost no correlation between the variables.

Heatmap for presence of correlations

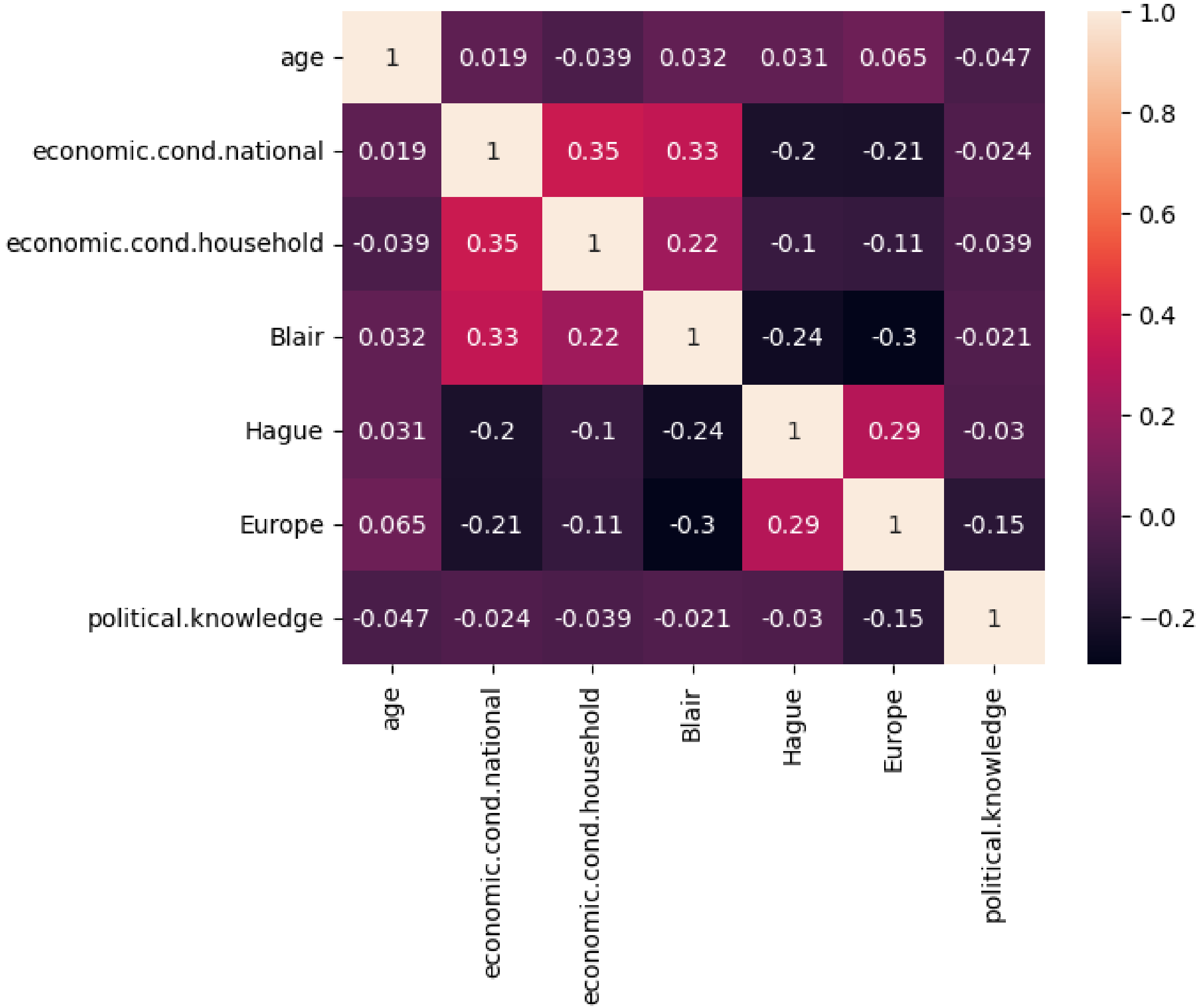


Fig.13 Heatmap showing correlation between different components

Observation

- There are some variables that are moderately positively correlated and some that are slightly negatively correlated.
- ‘economic.cond.national’ with ‘economic.cond.household’ have moderate positive correlation.
- ‘Blair' with 'economic.cond.national' and ‘economic.cond.household’ have moderate positive correlation.
- ‘Europe’ with ‘Hague’ have moderate positive correlation.
- 'Hague' with 'economic.cond.national' and 'Blair' have moderate negative correlation.
- 'Europe' with 'economic.cond.national' and 'Blair' have moderate negative correlation.

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

Encoded Data

Vote and gender being categorical columns have been encoded into 0s and 1s as shown below

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

Fig.14 Encoded Dataset

Encoded Data info is shown below. Vote and gender datatype has been converted from object to integer.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1517 entries, 0 to 1524
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   age                      1517 non-null   int64
 1   economic.cond.national   1517 non-null   int64
 2   economic.cond.household  1517 non-null   int64
 3   Blair                    1517 non-null   int64
 4   Hague                    1517 non-null   int64
 5   Europe                   1517 non-null   int64
 6   political.knowledge      1517 non-null   int64
 7   vote_Labour              1517 non-null   uint8
 8   gender_male              1517 non-null   uint8
dtypes: int64(7), uint8(2)
memory usage: 97.8 KB
```

Fig.15 Encoded information of dataset

Scaling

- Scaling is a fundamental preprocessing step in machine learning that involves adjusting the magnitude of input features.
- Its importance lies in ensuring that all features contribute equally to the learning process, preventing dominance by features with larger scales.
- Scaling promotes faster convergence of optimization algorithms, enhances model performance, and facilitates the correct functioning of distance-based algorithms.
- It also aids in feature interpretability, prevents numerical instability issues, and promotes consistent model performance.
- In this case, we have a lot of encoded, ordinal, categorical and continuous variables. So, we use the minmaxscaler technique to scale the data.

Scaled Dataset

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	vote_Labour	gender_male
0	0.275362	0.50	0.50	0.75	0.00	0.1	0.666667	1.0	0.0
1	0.173913	0.75	0.75	0.75	0.75	0.4	0.666667	1.0	1.0
2	0.159420	0.75	0.75	1.00	0.25	0.2	0.666667	1.0	1.0
3	0.000000	0.75	0.25	0.25	0.00	0.3	0.000000	1.0	0.0
4	0.246377	0.25	0.25	0.00	0.00	0.5	0.666667	1.0	1.0

Fig.16 Scaled dataset

Test-Train split

Our model will use all the variables and 'vote_Labour' is the target variable. The train-test split is a technique for evaluating the performance of a machine learning algorithm.

The data is divided into 2 subsets, training and testing set. Earlier, we have extracted the target variable 'vote_Labour' in a separate vector for subsets. Random state chosen as 1.

• **Training Set**: 70percent of data.
• **Testing Set**: 30 percent of the data

After splitting- the shape of the data

Table 1 Shape of Test Train split data

x_train	(1061,8)
y_train	(1061,)
x_test	(456,8)
y_test	(456,)

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

Logistic Regression Model

Classification Report for Train data

	precision	recall	f1-score	support
0.0	0.76	0.63	0.69	307
1.0	0.86	0.92	0.89	754
accuracy			0.83	1061
macro avg	0.81	0.77	0.79	1061
weighted avg	0.83	0.83	0.83	1061

Fig.17 LR-Classification report- Train data

Classification Report for Test data

	precision	recall	f1-score	support
0.0	0.76	0.71	0.73	153
1.0	0.86	0.89	0.87	303
accuracy			0.83	456
macro avg	0.81	0.80	0.80	456
weighted avg	0.82	0.83	0.83	456

Fig.18 LR- Classification report- Test data

Hyper-parameters

The model is constructed using the function LogisticRegression from the sklearn.linear_model library. Arguments passed for these functions are as below:

- **Penalty=none** No penalty is added to the model
- **random_state=1** This makes the model's output replicable. The model will always produce the same results when it has a definite value of random_state and if it has been given the same parameters and the same training data.
- **max_iter**=2000 It is the maximum number of iterations for the solvers to converge
- **tol=0.0001** This is the tolerance value for the stopping criteria.

Logistic Regression Model – Observation
Train data:
• Accuracy: 83.41%
• Precision: 86%
• Recall: 92%
• F1-Score: 89%

Test data:
• Accuracy: 82.68%
• Precision: 86%
• Recall: 89%
• F1-Score: 87%

Validness of Model

The model is not overfitted or underfitted as the scores of both the train and test dataset are similar. Training and Testing results shows that the model is excellent with good precision and recall values.

LDA (linear discriminant analysis)

Classification Report for Train data

	precision	recall	f1-score	support
0.0	0.74	0.65	0.69	307
1.0	0.86	0.91	0.89	754
accuracy			0.83	1061
macro avg	0.80	0.78	0.79	1061
weighted avg	0.83	0.83	0.83	1061

Fig.19 LDA-Classification report- Train data

Classification Report for Test data

	precision	recall	f1-score	support
0.0	0.77	0.73	0.74	153
1.0	0.86	0.89	0.88	303
accuracy			0.83	456
macro avg	0.82	0.81	0.81	456
weighted avg	0.83	0.83	0.83	456

Fig.20 LDA- Classification report- Test data

Hyper-parameters

The model is constructed using the function LinearDiscriminantAnalysis from the sklearn.discriminant_analysis library. We use default arguments for this model generation. The main default arguments are:

- **solver= svd** This is the algorithm to use in the optimization problem
- **tol=0.0001** This is the tolerance value for the stopping criteria
    -

LDA – Observation
Train data:
• Accuracy: 83.41%
• Precision: 86%
• Recall: 91%
• F1-Score: 89%

Test data:
• Accuracy: 83.33%
• Precision: 86%
• Recall: 89%
• F1-Score: 88%

Validness of Model

The model is not overfitted or underfitted as the scores for train and test dataset are similar. Training and Testing results shows that the model is excellent with good precision and recall values.

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

KNN Model

Classification Report for Train data

	precision	recall	f1-score	support
0.0	0.77	0.72	0.74	307
1.0	0.89	0.91	0.90	754
accuracy			0.86	1061
macro avg	0.83	0.82	0.82	1061
weighted avg	0.85	0.86	0.86	1061

Fig.21 KNN- Classification report- Train data

Classification Report for Test data

	precision	recall	f1-score	support
0.0	0.75	0.70	0.72	153
1.0	0.85	0.88	0.87	303
accuracy			0.82	456
macro avg	0.80	0.79	0.79	456
weighted avg	0.82	0.82	0.82	456

Fig.22 KNN- Classification report- Test data

Hyper-parameters

The model is constructed using the function KNeighborsClassifier from the sklearn.neighbors library. Arguments passed for these functions are as below:

- **n_neighbors=5** This is the number of neighbors that is used by default i.e. the k value
- **Penalty=none** No penalty is added to the model

KNN – Observation

Train data:
• Accuracy: 85.67%
• Precision: 89%
• Recall: 91%
• F1-Score: 90%

Test data:
• Accuracy: 82.02%
• Precision: 85%
• Recall: 88%
• F1-Score: 87%

Validness of Model

The scores for the train and test dataset are similar indicating that the generated model is not over-fitted. This KNN model has good accuracy and recall values.

Naïve Bayes Model

Classification Report for Train data

	precision	recall	f1-score	support
0.0	0.73	0.69	0.71	307
1.0	0.88	0.90	0.89	754
accuracy			0.84	1061
macro avg	0.80	0.79	0.80	1061
weighted avg	0.83	0.84	0.83	1061

Fig.23 NB- Classification report- Train data

Classification Report for Test data

	precision	recall	f1-score	support
0.0	0.74	0.73	0.73	153
1.0	0.87	0.87	0.87	303
accuracy			0.82	456
macro avg	0.80	0.80	0.80	456
weighted avg	0.82	0.82	0.82	456

Fig.24 NB- Classification report- Test data

Hyper-parameters

The model is constructed using the function GaussianNB from the sklearn.naive_bayes library. We use default arguments for this model generation.

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

Naïve Bayes – Observation

Train data:
• Accuracy: 83.51%
• Precision: 88%
• Recall: 90%
• F1-Score: 89%

Test data:
• Accuracy: 82.24%
• Precision: 87%
• Recall: 87%
• F1-Score: 87%

Validness of Model

The scores for the train and test dataset are similar indicating that the generated model is not over-fitted.

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

Logistic Regression Model (Tuned)

Below is the parameter grid which is given as the input for the Logistic Regression Model.

```
param_grid1 = {'penalty':['l2','none','l1','elasticnet'],
               'solver':['sag','lbfgs','saga','newton-cg','liblinear'],
               'tol':[0.001,0.0001,0.00001],
               'l1_ratio':[0.25,0.5,0.75],
               'max_iter':[100,1000,10000]}
```

After the GridSearchCV function execution is complete, below is the set of best selected parameters for the model

```
{'l1_ratio': 0.75,
 'max_iter': 100,
 'penalty': 'elasticnet',
 'solver': 'saga',
 'tol': 0.0001}
```

Logistic Regression Model (Tuned)– Observation

Train data:

• Accuracy: 83.03%
• Precision: 86%
• Recall: 91%
• F1-Score: 88%

Test data:

• Accuracy: 83.11%
• Precision: 86%
• Recall: 89%
• F1-Score: 87%

Comparison of Simple v/s tuned Model

We compare these results with the simple Logistic Regression model that was previously created in 1.4 Logistic Regression Model.

Table 2 Comparison of Simple v/s tuned-LR Model

Model	Dataset	Accuracy
Simple LR Model	Train Dataset	83.41%
	Test Dataset	82.68%
Tuned LR Model	Train Dataset	83.03%
	Test Dataset	83.11%

- The score for the train data has remained almost the same.
- The score for the test dataset has improved. Although the improvement is small, it shows that the GridSearchCV operation gave positive results.

LDA (Tuned)

Below is the parameter grid which is given as the input for the LDA Model.

```
param_grid2 = {'solver':[ 'svd', 'lsqr', 'eigen'],
               'tol':[0.001,0.0001,0.00001]}
```

After the GridSearchCV function execution is complete, below is the set of best selected parameters for the model

```
{'solver': 'svd', 'tol': 0.001}
```

LDA (Tuned)– Observation

Train data:
- Accuracy: 83.41%
- Precision: 86%
- Recall: 91%
- F1-Score: 89%

Test data:
- Accuracy: 83.33%
- Precision: 86%
- Recall: 89%
- F1-Score: 88%

Comparison of Simple v/s tuned Model

We compare these results with the simple Logistic Regression model that was previously created in 1.4 LDA Model.

Table 3 Comparison of Simple v/s tuned-LDA Model

Model	Dataset	Accuracy
Simple LDA Model	Train Dataset	83.41%
	Test Dataset	83.33%
Tuned LDA Model	Train Dataset	83.41%
	Test Dataset	83.33%

- The score for the train and test data has remained the same in both the simple and tuned Model.
- In this case, the model tuning operation did not generate any useful results.

KNN Model (Tuned)

Below is the parameter grid which is given as the input for the KNN Model.

```
leaf_size=list(range(1,50))
n_neighbors=list(range(1,30))
p=[1,2]
Hyper=dict(leaf_size=leaf_size,n_neighbors=n_neighbors,p=p)
```

After the GridSearchCV function execution is complete, below is the set of best selected parameters for the model

```
{'leaf_size': 14, 'n_neighbors': 28, 'p': 2}
```

KNN Model (Tuned)– Observation

Train data:
- Accuracy: 83.32%
- Precision: 86%
- Recall: 91%
- F1-Score: 89%

Test data:
- Accuracy: 82.90%
- Precision: 84%
- Recall: 91%
- F1-Score: 88%

Comparison of Simple v/s tuned Model

We compare these results with the simple KNN model that was previously created in 1.4 KNN Model.

Table 4 Comparison of Simple v/s tuned-KNN Model

Model	Dataset	Accuracy
Simple KNN Model	Train Dataset	85.67%
	Test Dataset	82.02%
Tuned KNN Model	Train Dataset	83.32%
	Test Dataset	82.90%

- The score for the train data has reduced a little.
- The score for the test dataset has improved. Although the improvement is small, it shows that the GridSearchCV operation gave positive results.

Naïve Bayes Model (Tuned)

Below is the parameter grid which is given as the input for the Naïve Bayes Model.

```
param_grid4 = {'var_smoothing': np.logspace(0,-9, num=1000)}
```

After the GridSearchCV function execution is complete, below is the set of best selected parameters for the model

```
{'var_smoothing': 0.1787525525904235}
```

Naïve Bayes Model (Tuned)– Observation

Train data:
• Accuracy: 83.13%
• Precision: 85%
• Recall: 92%
• F1-Score: 89%

Test data:
• Accuracy: 82.02%
• Precision: 84%
• Recall: 89%
• F1-Score: 87%

Comparison of Simple v/s tuned Model

We compare these results with the simple Naïve Bayes model that was previously created in 1.4 Naïve Bayes Model.

Table 5 Comparison of Simple v/s tuned- Naïve Bayes Model		
Model	**Dataset**	**Accuracy**
Simple Naïve Bayes Model	Train Dataset	83.51%
	Test Dataset	82.24%
Tuned Naïve Bayes Model	Train Dataset	83.13%
	Test Dataset	82.02%

- The score for the train and test data has remained almost the same in both the simple and tuned Model.
- In this case, the model tuning operation did not generate any useful results.

Bagging

Bagging Model uses base estimator as DecisionTreeClassifier.

Bagging Model – Observation

Train data:
• Accuracy: 100%
• Precision: 100%
• Recall: 100%
• F1-Score: 100%

Test data:
• Accuracy: 82.02%
• Precision: 86%
• Recall: 88%
• F1-Score: 87%

The dataset shows overfitting as the difference between accuracy of train and test is more than 10%. Hence, the dataset needs to be tuned.

Bagging (Tuned)

Below is the parameter grid which is given as the input for the Bagging Model.

```
param_grid5 = {
    'min_samples_split' : [30,50,70,100],
    'min_samples_leaf':[15,25,35,50],
    'max_depth':[5,10,15,20],
    'random_state' : [0]
}
```

After the GridSearchCV function execution is complete, below is the set of best selected parameters for the model.

```
{'max_depth': 10,
 'min_samples_leaf': 15,
 'min_samples_split': 50,
 'random_state': 0}
```

Bagging Model (Tuned)– Observation

Train data:
• Accuracy: 83.98%
• Precision: 90%
• Recall: 87%
• F1-Score: 89%

Test data:
• Accuracy: 79.61%

• Precision: 86%
• Recall: 83%
• F1-Score: 84%

Comparison of Simple v/s tuned Model

We compare these results with the simple Bagging model that is created above.

Table 6 Comparison of Simple v/s tuned Bagging Model

Model	Dataset	Accuracy
Simple Bagging Model	Train Dataset	100%
	Test Dataset	82.02%
Tuned Bagging Model	Train Dataset	83.98%
	Test Dataset	79.61%

- The scores for the train data indicates that the tuned data is no longer overfitted.
- The simple model has performed slightly better in test data but as the tuned data is not overfitted, hence it is better than the simple model.

Random Forest Model

Random forest is an extension of the bagging that also randomly selects subsets of features used in each data sample.

Random Forest Model – Observation

Train data:
- Accuracy: 100%
- Precision: 100%
- Recall: 100%
- F1-Score: 100%

Test data:
- Accuracy: 83.11%
- Precision: 85%
- Recall: 90%
- F1-Score: 88%

The dataset shows overfitting as the difference between accuracy of train and test is more than 10%. Hence, the dataset needs to be tuned.

Random Forest Model (Tuned)

Below is the parameter grid which is given as the input for the Random Forest Model.

```
param_grid8 = {
    'min_samples_split' : [30,50,70,100],
    'min_samples_leaf':[15,25,35,50],
    'max_depth':[5,10,15,20],
    'random_state' : [0]
}
```

After the GridSearchCV function execution is complete, below is the set of best selected parameters for the model.

```
RandomForestClassifier(max_depth=10, min_samples_leaf=15, min_samples_split=30,
                       random_state=0)
```

Random Forest Model (Tuned)– Observation

Train data:
- Accuracy: 85.77%
- Precision: 87%
- Recall: 94%
- F1-Score: 90%

Test data:
- Accuracy: 81.80%
- Precision: 82%

• Recall: 92%
• F1-Score: 87%

Comparison of Simple v/s tuned Model

We compare these results with the simple Random Forest model that is created above.

Table 7 Comparison of Simple v/s tuned-Random Forest Model

Model	Dataset	Accuracy
Simple Random Forest Model	Train Dataset	100%
	Test Dataset	83.11%
Tuned Random Forest Model	Train Dataset	85.77%
	Test Dataset	81.80%

- The scores for the train data indicates that the tuned data is no longer overfitted.
- The simple model has performed slightly better in test data but as the tuned data is not overfitted, hence it is better than the simple model.

Feature Importance

	Imp
age	0.212777
economic.cond.national	0.092387
economic.cond.household	0.081470
Blair	0.133116
Hague	0.178677
Europe	0.188021
political.knowledge	0.077852
gender_male	0.035700

Fig.25 RF Feature Importance

- Feature importance scores are used to determine the relative importance of each feature in a dataset when building a machine learning model.
- Here age is the most impactful feature with highest score of 0.212

Ada Boost Model

It works on the principle of learners growing sequentially i.e. except for the first learner, each subsequent learner is grown from previously grown learners.

Ada Boost Model – Observation

Train data:
• Accuracy: 84.26%
• Precision: 87%
• Recall: 91%
• F1-Score: 89%

Test data:
• Accuracy: 82.02%
• Precision: 86%
• Recall: 87%
• F1-Score: 87%

The scores for the train and test dataset are similar indicating that the generated model is not over-fitted.

Ada Boost Model (Tuned)

Below is the parameter grid which is given as the input for the Ada Boost Model.

```
param_grid6 = {
    'n_estimators' : [100,500,1000],
    'learning_rate' : [0.1,0.01,0.001],
    'algorithm' : ['SAMME', 'SAMME.R']
}
```

After the GridSearchCV function execution is complete, below is the set of best selected parameters for the model.

```
AdaBoostClassifier(learning_rate=0.01, n_estimators=1000)
```

Ada Boost Model (Tuned)– Observation

Train data:
• Accuracy: 83.70%
• Precision: 85%
• Recall: 93%
• F1-Score: 89%

Test data:
• Accuracy: 80.92%

• Precision: 83%
• Recall: 89%
• F1-Score: 86%

Comparison of Simple v/s tuned Model

We compare these results with the simple Ada Boost model that is created above.

Table 8 Comparison of Simple v/s tuned-Ada Boost Model

Model	Dataset	Accuracy
Simple Ada Boost Model	Train Dataset	84.26%
	Test Dataset	82.02%
Tuned Ada Boost Model	Train Dataset	83.70%
	Test Dataset	80.92%

- The score for the train data and the test data has reduced slightly.
- In this case, the model tuning operation did not generate any useful results.

Gradient Boost Model

Gradient boosting is a stage-wise additive model that generates learners during the learning process.

Gradient Boost Model – Observation

Train data:
• Accuracy: 89.26%
• Precision: 91%
• Recall: 94%
• F1-Score: 93%

Test data:
• Accuracy: 83.33%
• Precision: 85%
• Recall: 91%
• F1-Score: 88%

The scores for the train and test dataset indicate that the generated model is not over-fitted.

Gradient Boost Model (Tuned)

Below is the parameter grid which is given as the input for the Gradient Boost Model.

```
param_grid7 = { 'n_estimators':[100, 250, 500, 600, 700],
                'learning_rate': [0.01, 0.1, 0.2, 0.3, 0.35, 0.5, 0.7, 0.8, 1]}
```

After the GridSearchCV function execution is complete, below is the set of best selected parameters for the model.

```
{'learning_rate': 0.01, 'n_estimators': 600}
```

Gradient Boost Model (Tuned)– Observation

Train data:
• Accuracy: 88.03%
• Precision: 90%
• Recall: 94%
• F1-Score: 92%

Test data:
• Accuracy: 83.11%
• Precision: 85%
• Recall: 91%
• F1-Score: 88%

Comparison of Simple v/s tuned Model

We compare these results with the simple Gradient Boost model that is created above.

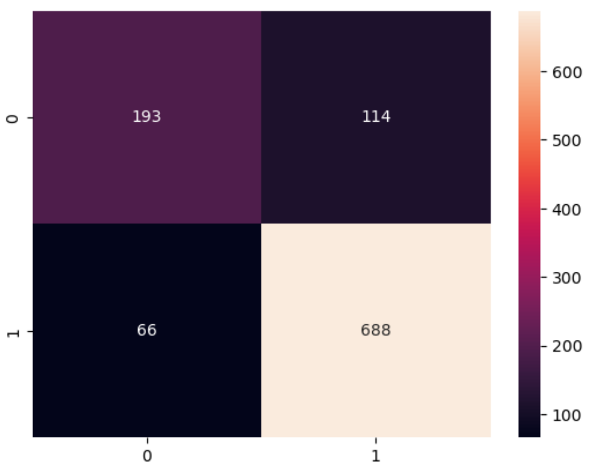
Table 9 Comparison of Simple v/s tuned-Gradient Boost Model

Model	Dataset	Accuracy
Simple Gradient_Boost Model	Train Dataset	89.26%
	Test Dataset	83.33%
Tuned Gradient_Boost Model	Train Dataset	88.03%
	Test Dataset	83.11%

- The score for the train and test data has almost remained the same in both the simple and tuned Model.
- In this case, the model tuning operation did not generate any useful results.

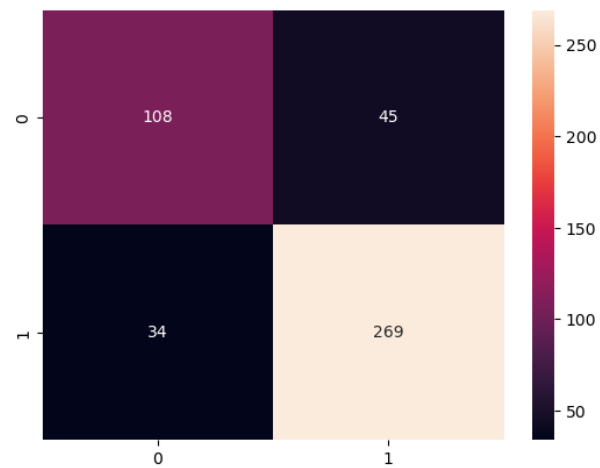
1.7 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model. Final Model: Compare the models and write inference which model is best/optimized.

<u>**Linear Regression Model**</u>



	precision	recall	f1-score	support
0.0	0.74	0.65	0.69	307
1.0	0.86	0.91	0.89	754
accuracy			0.83	1061
macro avg	0.80	0.78	0.79	1061
weighted avg	0.83	0.83	0.83	1061

Train Data



	precision	recall	f1-score	support
0.0	0.77	0.73	0.74	153
1.0	0.86	0.89	0.88	303
accuracy			0.83	456
macro avg	0.82	0.81	0.81	456
weighted avg	0.83	0.83	0.83	456

Test Data

Fig 26 Confusion matrix and classification report of LR Model

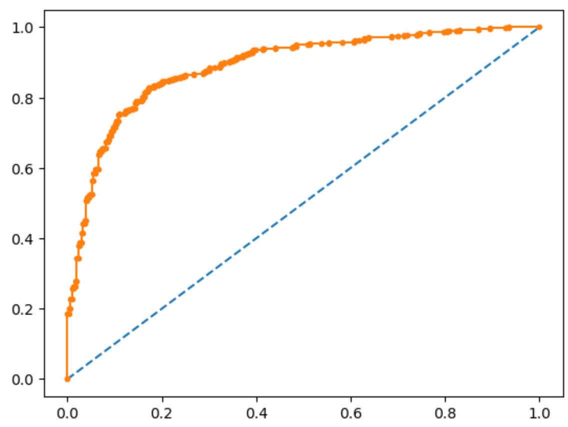
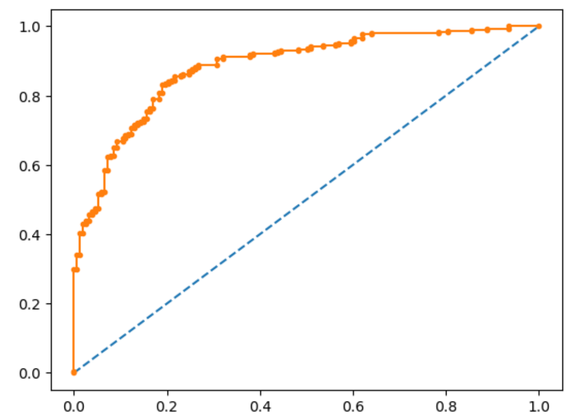
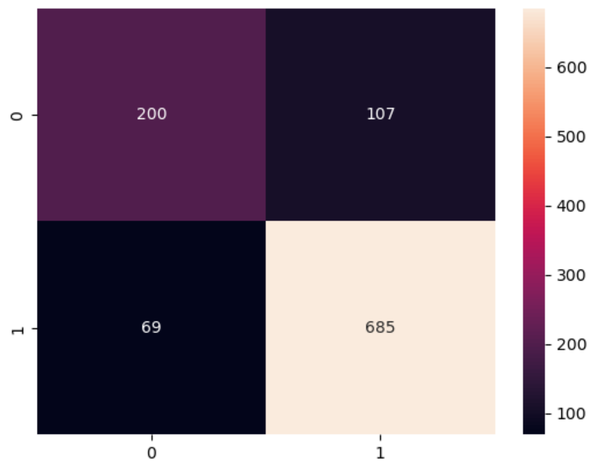
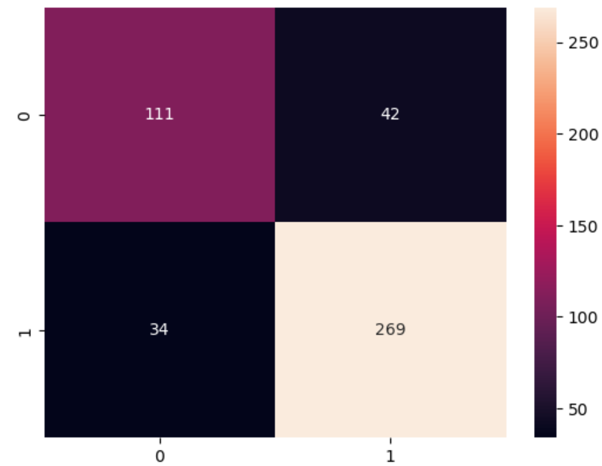
	
AUC:0.890	**AUC:0.884**
Train Data	**Test Data**

Fig 27 Roc-auc curve LR Model

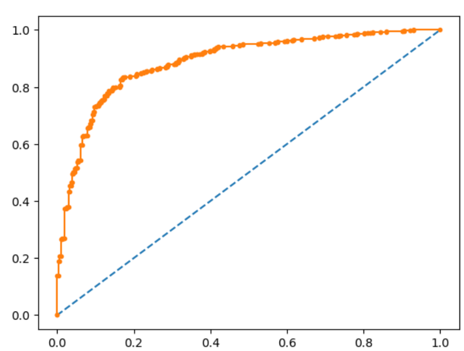
LDA Model



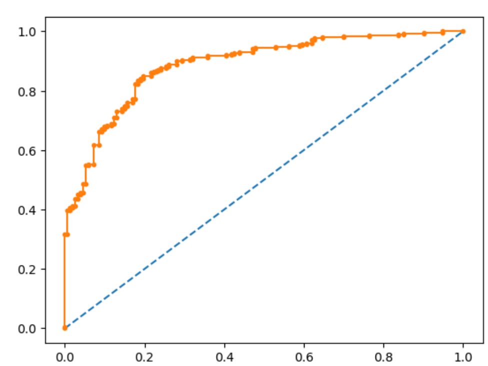
	precision	recall	f1-score	support
0.0	0.74	0.65	0.69	307
1.0	0.86	0.91	0.89	754
accuracy			0.83	1061
macro avg	0.80	0.78	0.79	1061
weighted avg	0.83	0.83	0.83	1061



	precision	recall	f1-score	support
0.0	0.77	0.73	0.74	153
1.0	0.86	0.89	0.88	303
accuracy			0.83	456
macro avg	0.82	0.81	0.81	456
weighted avg	0.83	0.83	0.83	456



AUC:0.889

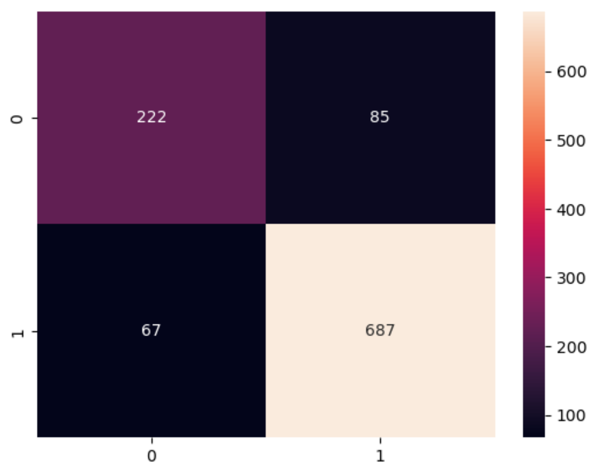


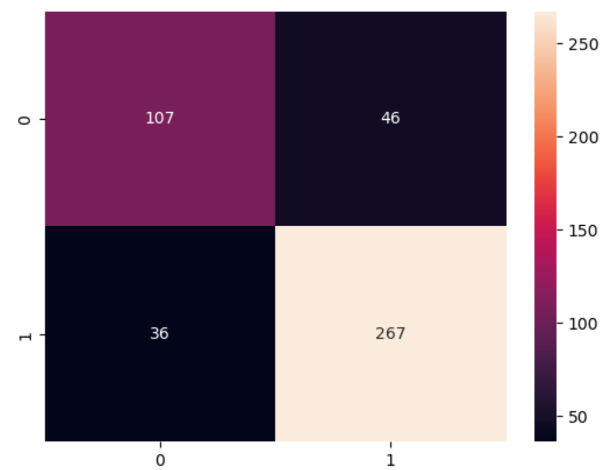
AUC:0.888

Train Data	Test Data

Fig 28 Confusion matrix, Classification report and AUC-ROC curve-LDA Model

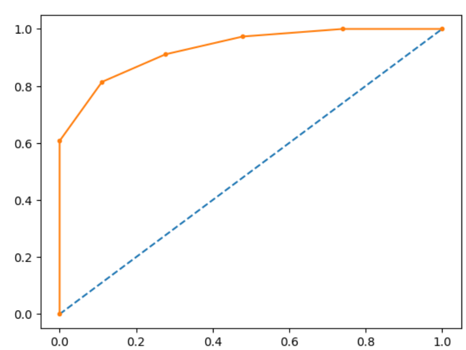
KNN Model



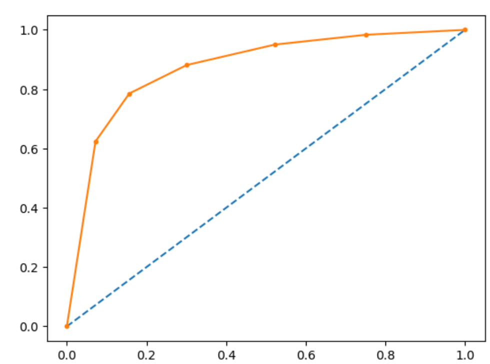


	precision	recall	f1-score	support
0.0	0.77	0.72	0.74	307
1.0	0.89	0.91	0.90	754
accuracy			0.86	1061
macro avg	0.83	0.82	0.82	1061
weighted avg	0.85	0.86	0.86	1061

	precision	recall	f1-score	support
0.0	0.75	0.70	0.72	153
1.0	0.85	0.88	0.87	303
accuracy			0.82	456
macro avg	0.80	0.79	0.79	456
weighted avg	0.82	0.82	0.82	456



AUC:0.930

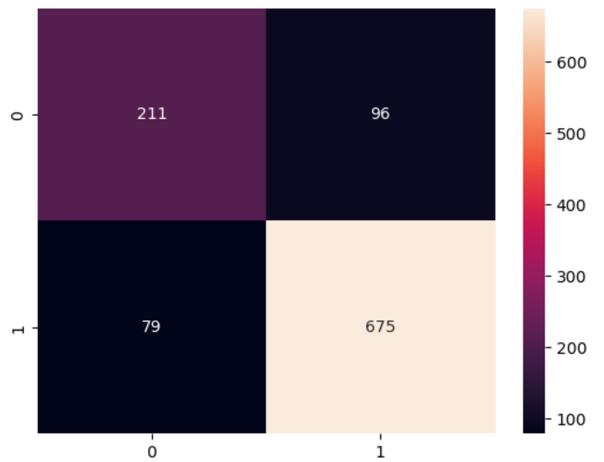


AUC:0.873

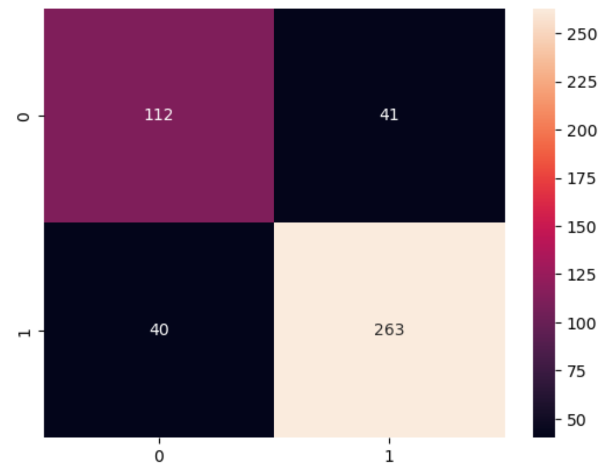
Train Data	**Test Data**

Fig 29 Confusion matrix, Classification report and AUC-ROC curve-KNN Model

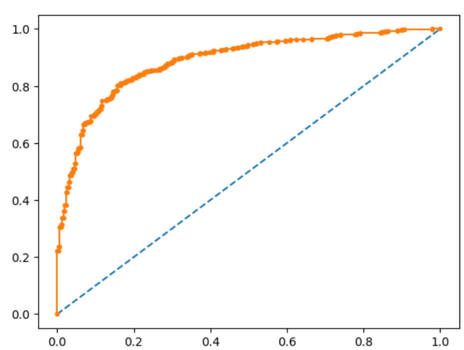
Naïve Bayes Model



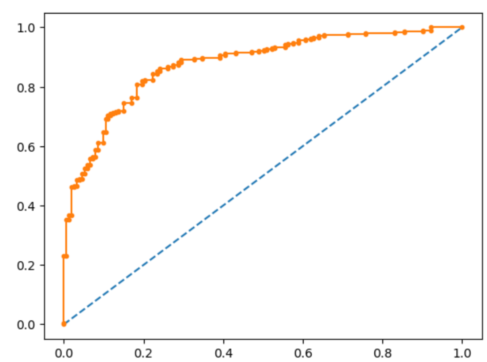
	precision	recall	f1-score	support
0.0	0.73	0.69	0.71	307
1.0	0.88	0.90	0.89	754
accuracy			0.84	1061
macro avg	0.80	0.79	0.80	1061
weighted avg	0.83	0.84	0.83	1061



	precision	recall	f1-score	support
0.0	0.74	0.73	0.73	153
1.0	0.87	0.87	0.87	303
accuracy			0.82	456
macro avg	0.80	0.80	0.80	456
weighted avg	0.82	0.82	0.82	456



AUC:0.888

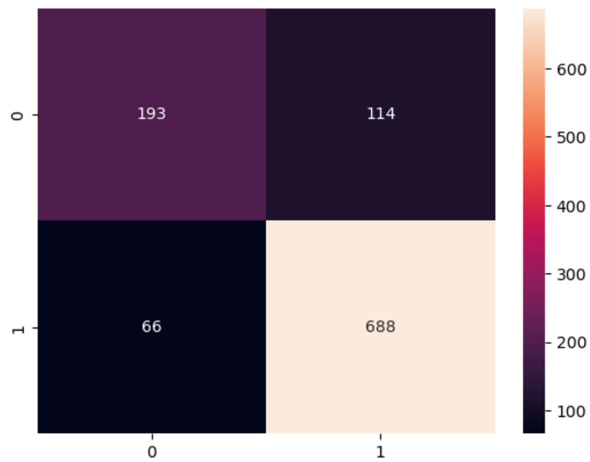


AUC:0.876

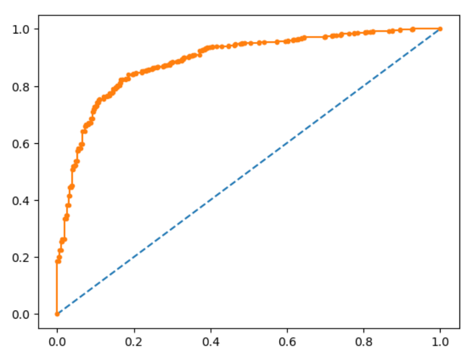
Train Data	**Test Data**

Fig 30 Confusion matrix, Classification report and AUC-ROC curve of Naïve Bayes Model

Linear Regression Model (Tuned)

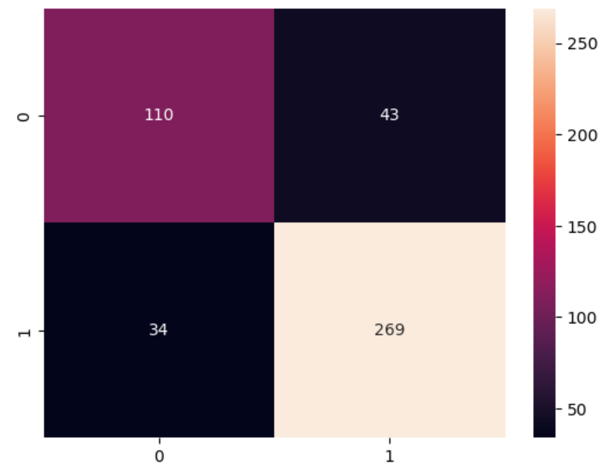


	precision	recall	f1-score	support
0.0	0.75	0.63	0.68	307
1.0	0.86	0.91	0.88	754
accuracy			0.83	1061
macro avg	0.80	0.77	0.78	1061
weighted avg	0.83	0.83	0.83	1061

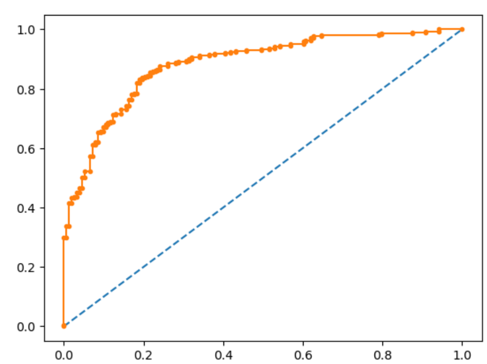


AUC:0.890

Train Data



	precision	recall	f1-score	support
0.0	0.76	0.72	0.74	153
1.0	0.86	0.89	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

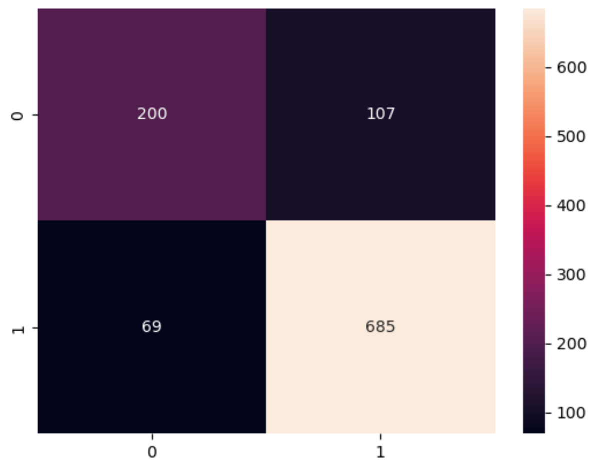


AUC:0.884

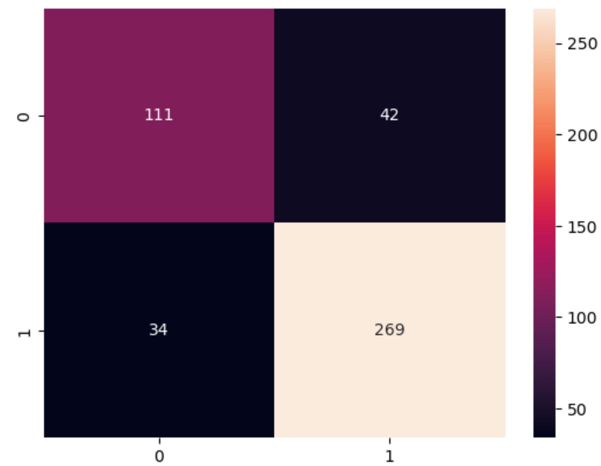
Test Data

Fig 31 Confusion matrix, Classification report and AUC-ROC curve of LR Model (Tuned)

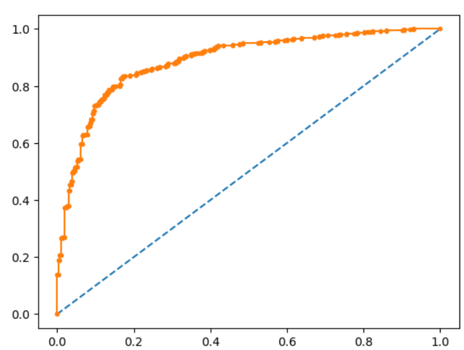
LDA(Tuned)



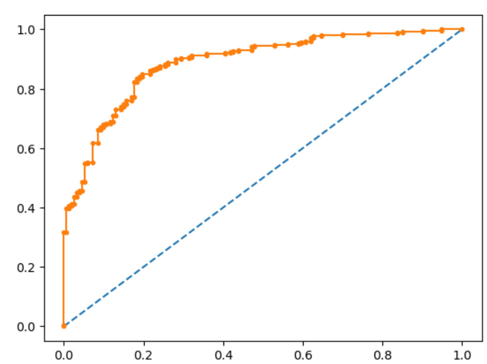
	precision	recall	f1-score	support
0.0	0.74	0.65	0.69	307
1.0	0.86	0.91	0.89	754
accuracy			0.83	1061
macro avg	0.80	0.78	0.79	1061
weighted avg	0.83	0.83	0.83	1061



	precision	recall	f1-score	support
0.0	0.77	0.73	0.74	153
1.0	0.86	0.89	0.88	303
accuracy			0.83	456
macro avg	0.82	0.81	0.81	456
weighted avg	0.83	0.83	0.83	456



AUC:0.889



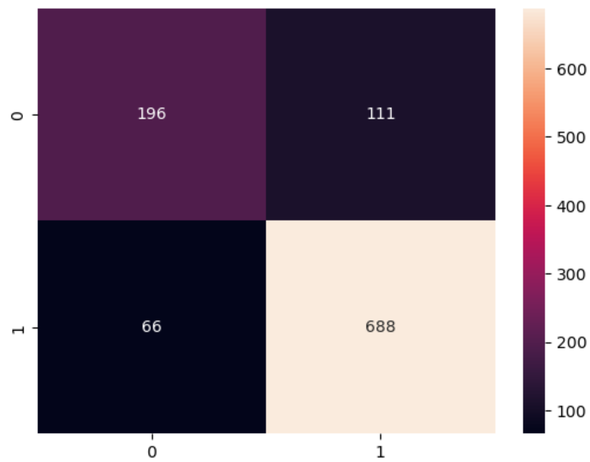
AUC:0.888

Train Data | **Test Data**

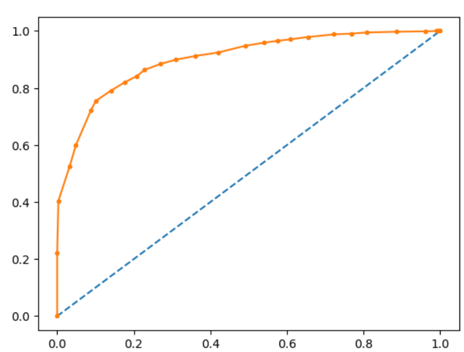
Fig 32 Confusion matrix, Classification report and AUC-ROC curve of LDA(Tuned)

KNN Model (Tuned)

Train Data

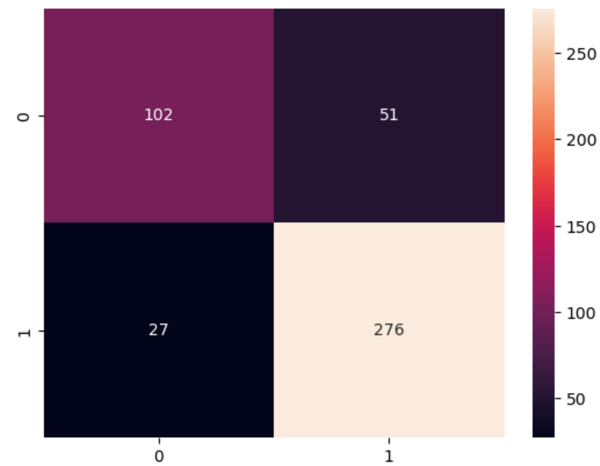


	precision	recall	f1-score	support
0.0	0.75	0.64	0.69	307
1.0	0.86	0.91	0.89	754
accuracy			0.83	1061
macro avg	0.80	0.78	0.79	1061
weighted avg	0.83	0.83	0.83	1061

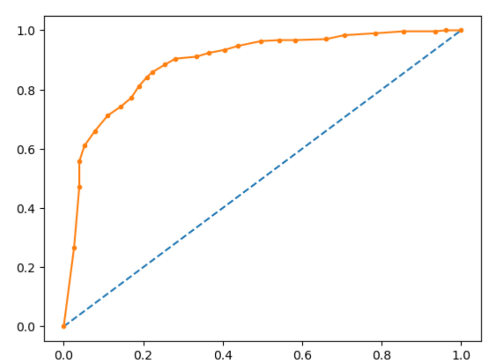


AUC:0.902

Test Data



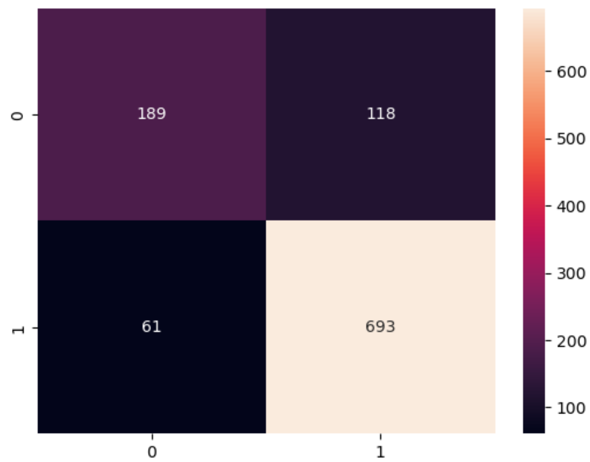
	precision	recall	f1-score	support
0.0	0.79	0.67	0.72	153
1.0	0.84	0.91	0.88	303
accuracy			0.83	456
macro avg	0.82	0.79	0.80	456
weighted avg	0.83	0.83	0.82	456



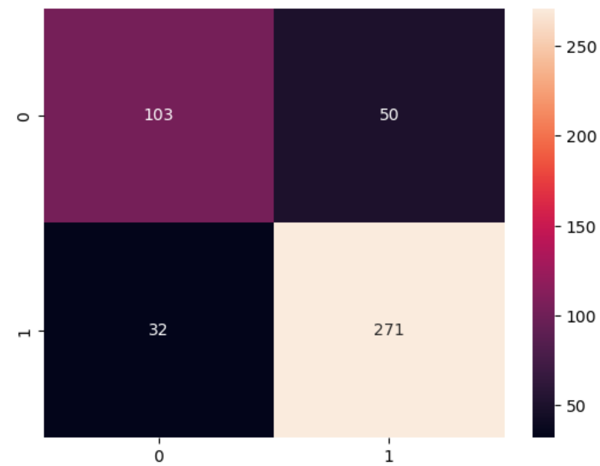
AUC:0.889

Fig 33 Confusion matrix, Classification report and AUC-ROC curve of KNN Model (Tuned)

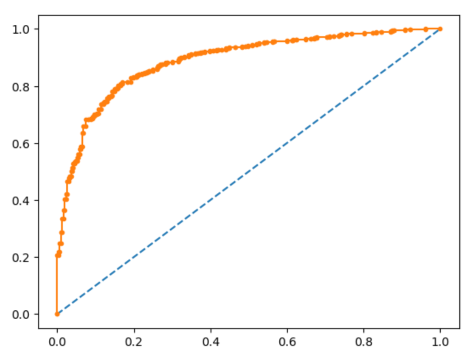
Naïve Bayes Model (Tuned)



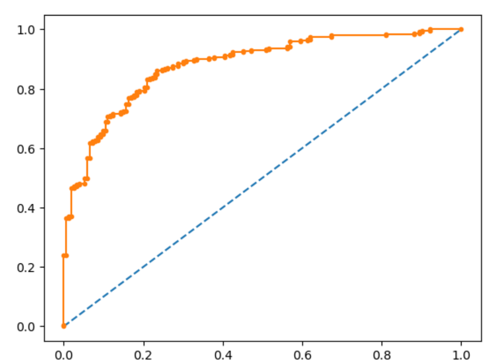
	precision	recall	f1-score	support
0.0	0.76	0.62	0.68	307
1.0	0.85	0.92	0.89	754
accuracy			0.83	1061
macro avg	0.81	0.77	0.78	1061
weighted avg	0.83	0.83	0.83	1061



	precision	recall	f1-score	support
0.0	0.76	0.67	0.72	153
1.0	0.84	0.89	0.87	303
accuracy			0.82	456
macro avg	0.80	0.78	0.79	456
weighted avg	0.82	0.82	0.82	456



AUC:0.888

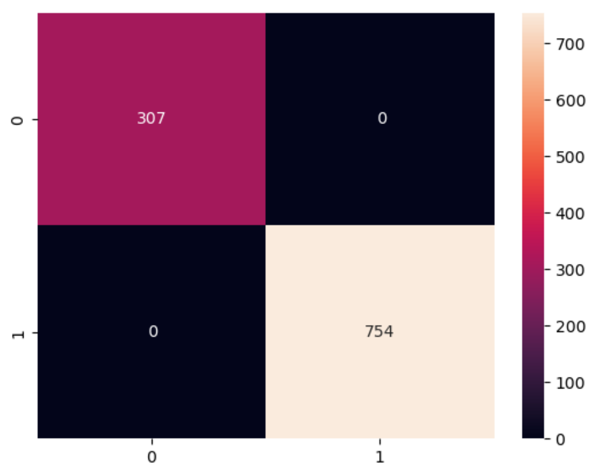


AUC:0.881

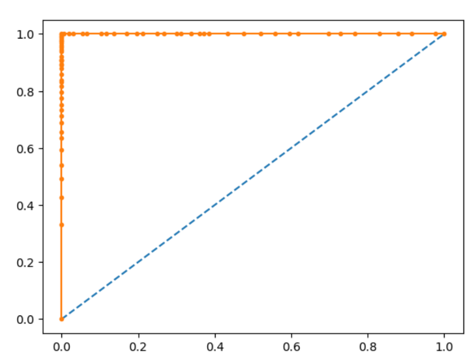
Train Data	**Test Data**

Fig 34 Confusion matrix, Classification report and AUC-ROC curve of Naïve Bayes Model (Tuned)

Bagging Model

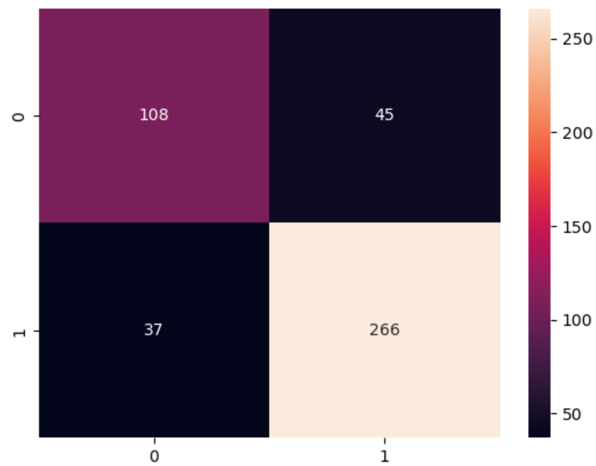


	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	307
1.0	1.00	1.00	1.00	754
accuracy			1.00	1061
macro avg	1.00	1.00	1.00	1061
weighted avg	1.00	1.00	1.00	1061

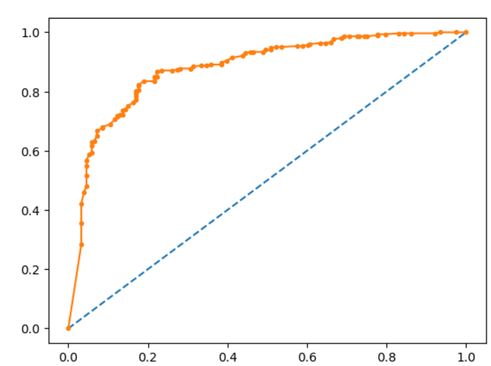


AUC:1.0

Train Data



	precision	recall	f1-score	support
0.0	0.74	0.71	0.72	153
1.0	0.86	0.88	0.87	303
accuracy			0.82	456
macro avg	0.80	0.79	0.80	456
weighted avg	0.82	0.82	0.82	456

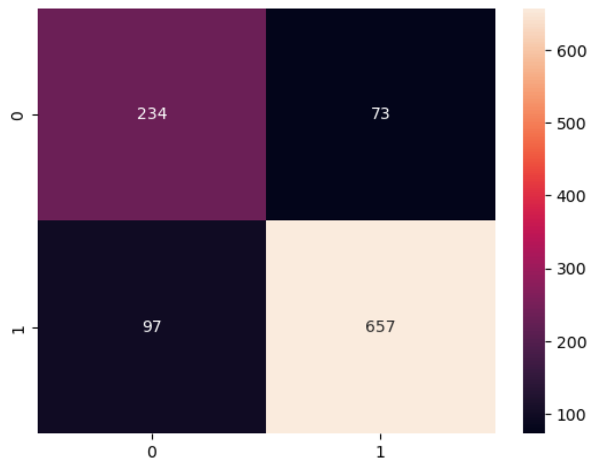


AUC:0.880

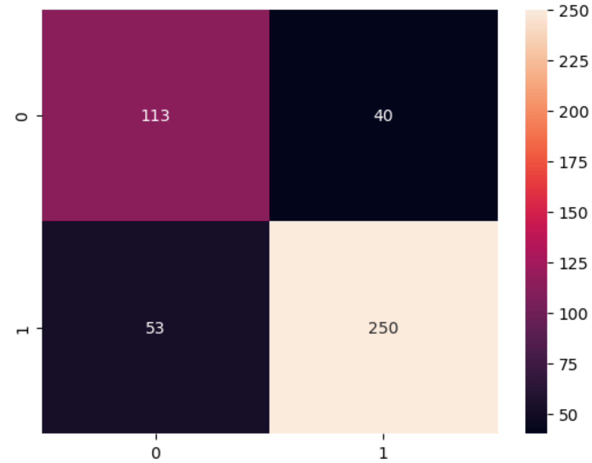
Test Data

Fig 35 Confusion matrix, Classification report and AUC-ROC curve of Bagging Model

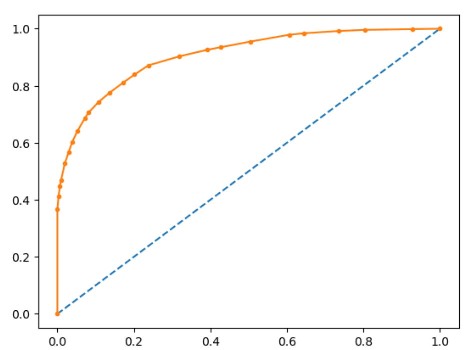
Bagging Model (Tuned)



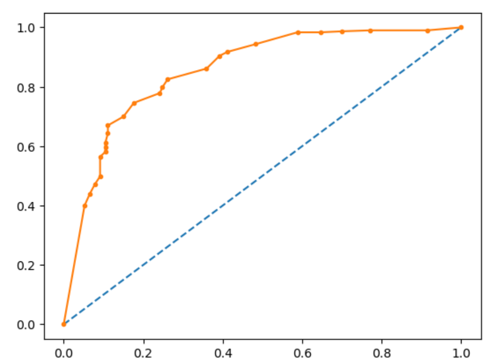
	precision	recall	f1-score	support
0.0	0.71	0.76	0.73	307
1.0	0.90	0.87	0.89	754
accuracy			0.84	1061
macro avg	0.80	0.82	0.81	1061
weighted avg	0.84	0.84	0.84	1061



	precision	recall	f1-score	support
0.0	0.68	0.74	0.71	153
1.0	0.86	0.83	0.84	303
accuracy			0.80	456
macro avg	0.77	0.78	0.78	456
weighted avg	0.80	0.80	0.80	456



AUC:0.907

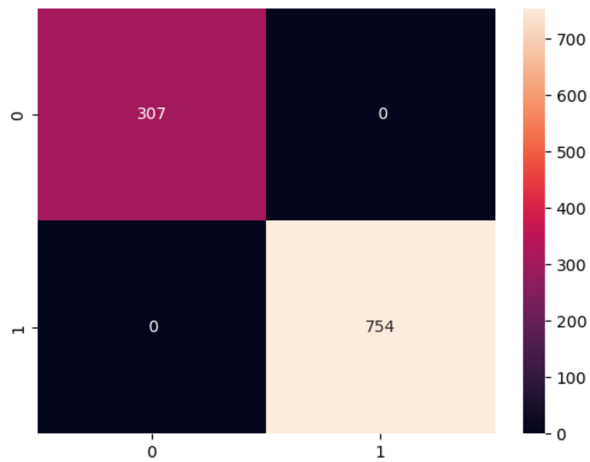


AUC:0.856

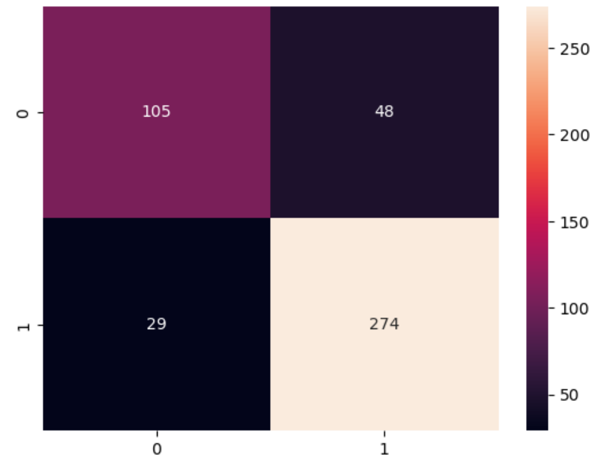
Train Data	Test Data

Fig 36 Confusion matrix, Classification report and AUC-ROC curve of Bagging Model (Tuned)

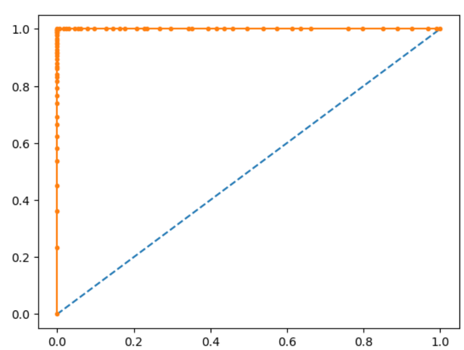
<u>Random Forest Model</u>



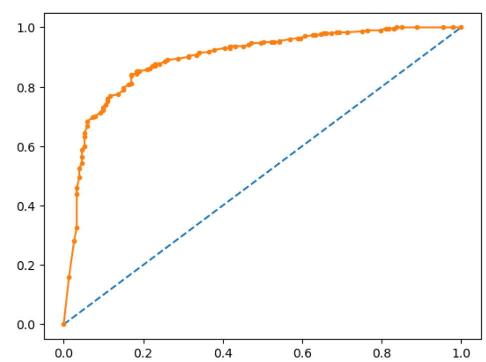
	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	307
1.0	1.00	1.00	1.00	754
accuracy			1.00	1061
macro avg	1.00	1.00	1.00	1061
weighted avg	1.00	1.00	1.00	1061



	precision	recall	f1-score	support
0.0	0.78	0.69	0.73	153
1.0	0.85	0.90	0.88	303
accuracy			0.83	456
macro avg	0.82	0.80	0.80	456
weighted avg	0.83	0.83	0.83	456



AUC:1.0



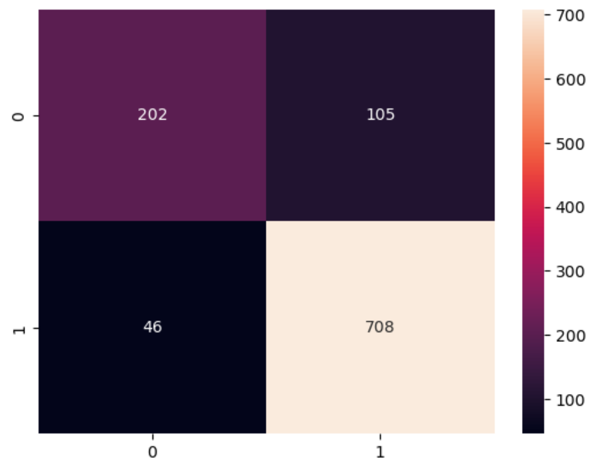
AUC:0.895

Train Data	**Test Data**

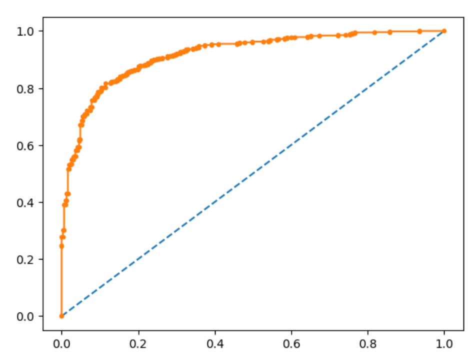
Fig 37 Confusion matrix, Classification report and AUC-ROC curve of Random Forest

Random Forest Model (Tuned)

Train Data

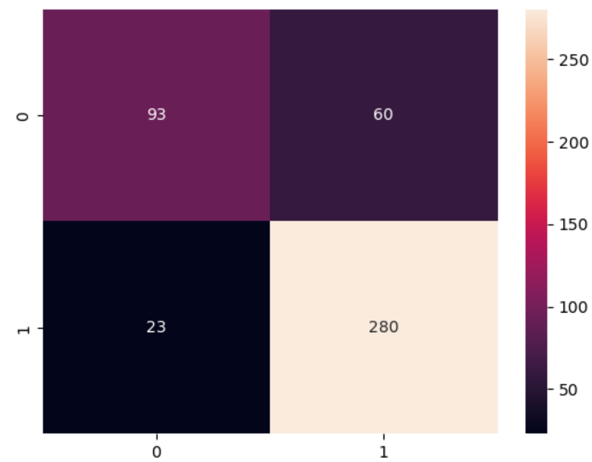


	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	307
1.0	1.00	1.00	1.00	754
accuracy			1.00	1061
macro avg	1.00	1.00	1.00	1061
weighted avg	1.00	1.00	1.00	1061

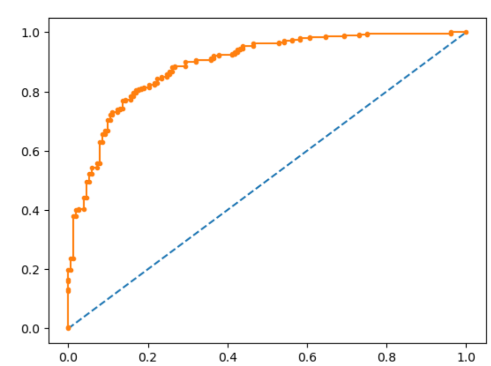


AUC:0.918

Test Data



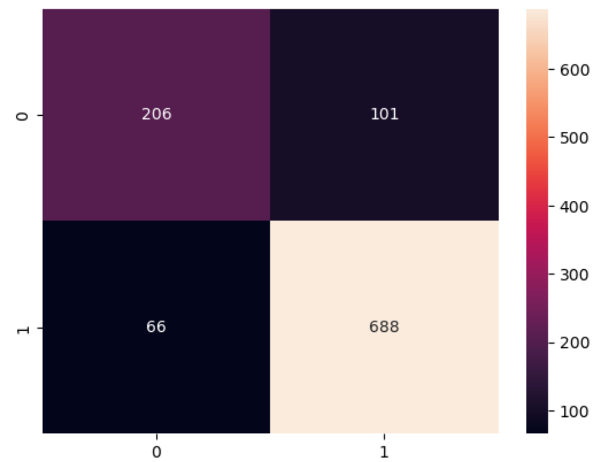
	precision	recall	f1-score	support
0.0	0.80	0.61	0.69	153
1.0	0.82	0.92	0.87	303
accuracy			0.82	456
macro avg	0.81	0.77	0.78	456
weighted avg	0.82	0.82	0.81	456



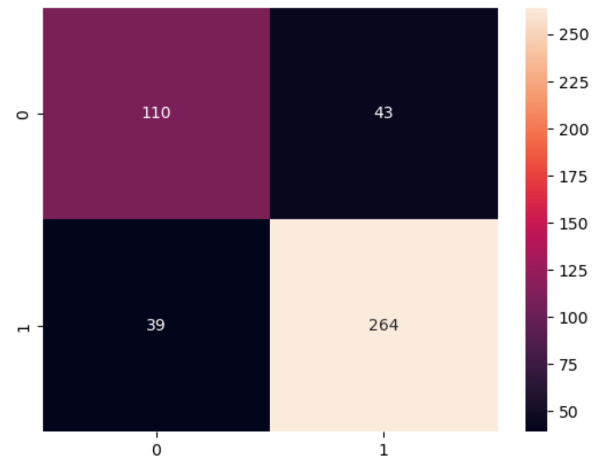
AUC:0.891

Fig 38 Confusion matrix, Classification report and AUC-ROC curve of Random Forest (Tuned)

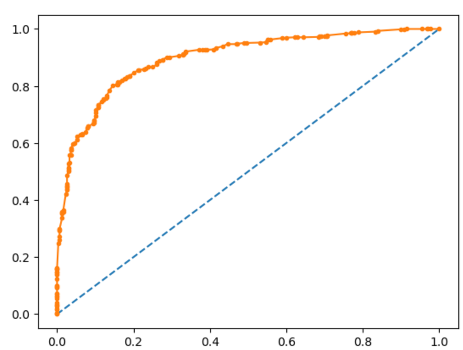
Ada Boost Model



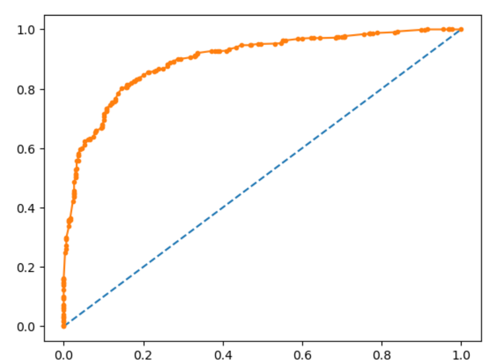
	precision	recall	f1-score	support
0.0	0.76	0.67	0.71	307
1.0	0.87	0.91	0.89	754
accuracy			0.84	1061
macro avg	0.81	0.79	0.80	1061
weighted avg	0.84	0.84	0.84	1061



	precision	recall	f1-score	support
0.0	0.74	0.72	0.73	153
1.0	0.86	0.87	0.87	303
accuracy			0.82	456
macro avg	0.80	0.80	0.80	456
weighted avg	0.82	0.82	0.82	456



AUC:0.878

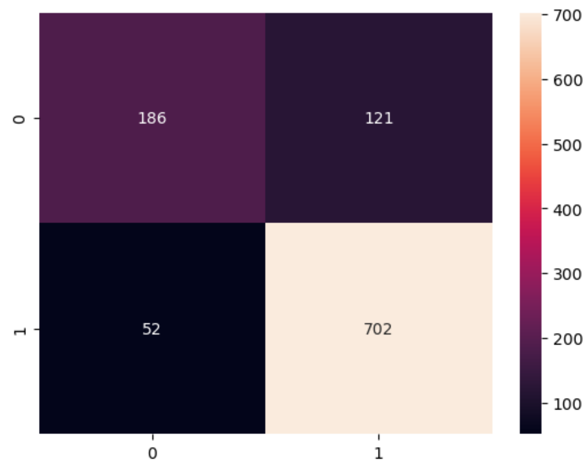


AUC:0.898

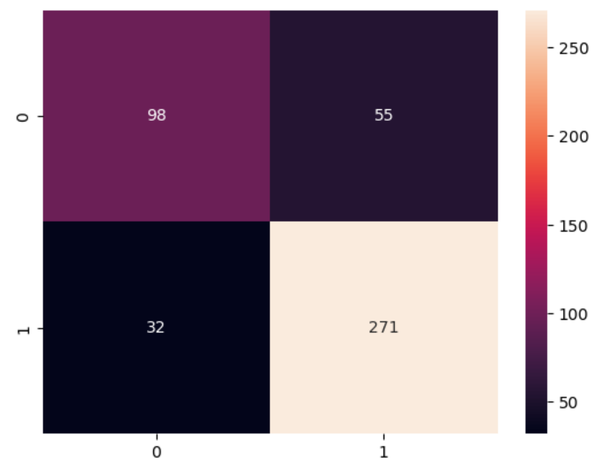
Train Data | **Test Data**

Fig 39 Confusion matrix, Classification report and AUC-ROC curve of Ada Boost Model

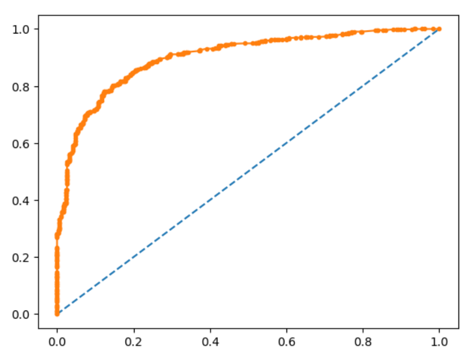
Ada Boost Model (Tuned)



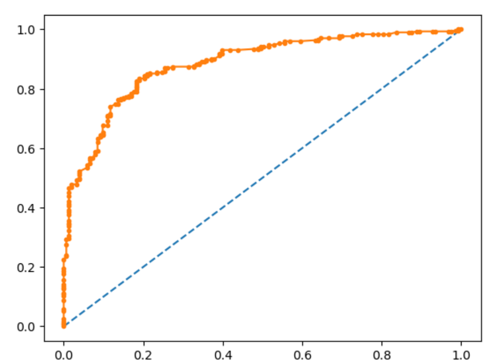
	precision	recall	f1-score	support
0.0	0.78	0.61	0.68	307
1.0	0.85	0.93	0.89	754
accuracy			0.84	1061
macro avg	0.82	0.77	0.79	1061
weighted avg	0.83	0.84	0.83	1061



	precision	recall	f1-score	support
0.0	0.75	0.64	0.69	153
1.0	0.83	0.89	0.86	303
accuracy			0.81	456
macro avg	0.79	0.77	0.78	456
weighted avg	0.81	0.81	0.80	456



AUC:0.902

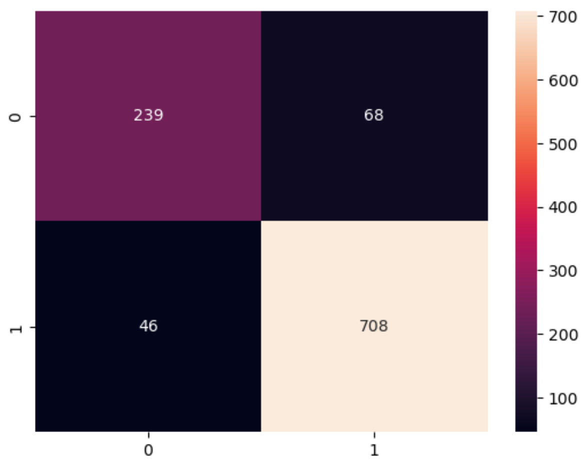


AUC:0.884

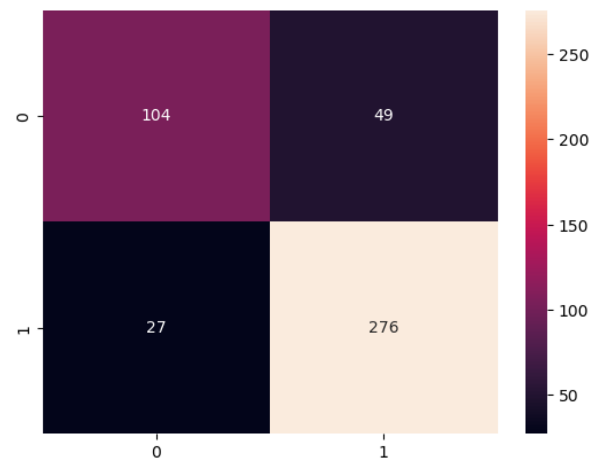
Train Data	**Test Data**

Fig 40 Confusion matrix, Classification report and AUC-ROC curve of Ada Boost Model (Tuned)

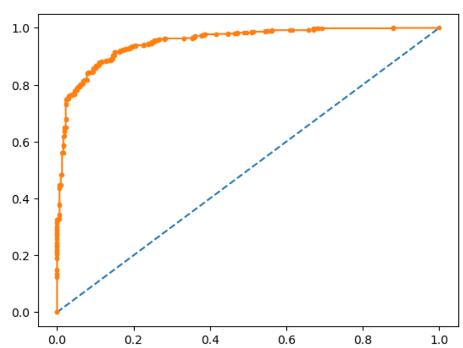
Gradient Boost Model



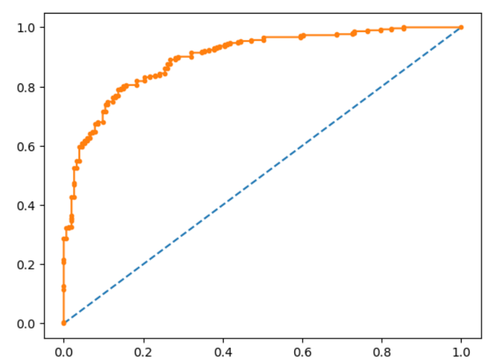
	precision	recall	f1-score	support
0.0	0.84	0.78	0.81	307
1.0	0.91	0.94	0.93	754
accuracy			0.89	1061
macro avg	0.88	0.86	0.87	1061
weighted avg	0.89	0.89	0.89	1061



	precision	recall	f1-score	support
0.0	0.79	0.68	0.73	153
1.0	0.85	0.91	0.88	303
accuracy			0.83	456
macro avg	0.82	0.80	0.81	456
weighted avg	0.83	0.83	0.83	456



AUC:0.951

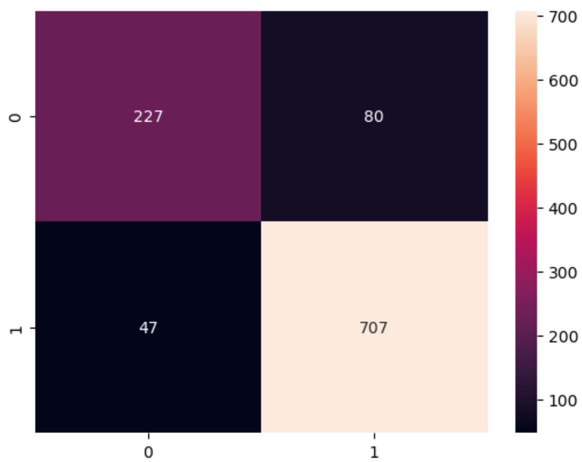


AUC:0.899

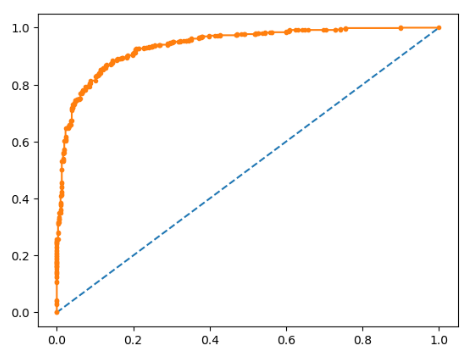
Train Data | **Test Data**

Fig 41 Confusion matrix, Classification report and AUC-ROC curve of Gradient Boost Model

Gradient Boost Model (Tuned)

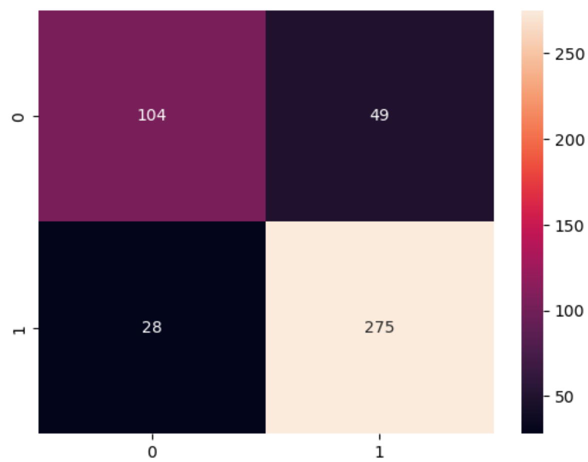


	precision	recall	f1-score	support
0.0	0.83	0.74	0.78	307
1.0	0.90	0.94	0.92	754
accuracy			0.88	1061
macro avg	0.86	0.84	0.85	1061
weighted avg	0.88	0.88	0.88	1061

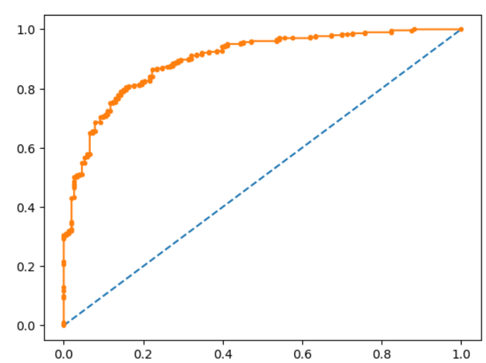


AUC:0.939

Train Data



	precision	recall	f1-score	support
0.0	0.79	0.68	0.73	153
1.0	0.85	0.91	0.88	303
accuracy			0.83	456
macro avg	0.82	0.79	0.80	456
weighted avg	0.83	0.83	0.83	456



AUC:0.898

Test Data

Fig 42 Confusion matrix, Classification report and AUC-ROC curve of Gradient Boost Model (Tuned)

Comparison of train data for all the Models

Models	Accuracy (%)	Precision (%)	Recall (%)	F1 score (%)	AUC Score (%)
LR- Simple	83.41	86	92	89	89
LR- Tuned	83.11	86	91	88	89
LDA -Simple	83.41	86	91	89	88.9
LDA-Tuned	83.41	86	91	89	88.9
KNN -Simple	85.67	89	91	90	93
KNN- Tuned	83.32	86	91	89	90.2
Naïve Bayes - Simple	83.50	88	90	89	88.8
Naïve Bayes - Tuned	83.13	85	92	89	88.8
Bagging – Simple	100	100	100	100	100
Bagging- Tuned	83.98	90	87	89	90.7
Random Forest- Simple	100	100	100	100	100
Random Forest- Tuned	85.77	87	94	90	91.8
Ada Boost - Simple	84.26	87	91	89	89.8
Ada Boost - Tuned	83.69	85	93	89	90.2
Gradient Boost- Simple	89.26	91	94	93	95.1
Gradient Boost- Tuned	88.03	90	94	92	93.9

Table 10 Comparison of train data for all the Models

<u>Comparison of test data for all the Models</u>

Models	Accuracy (%)	Precision (%)	Recall (%)	F1 score (%)	AUC Score (%)
LR- Simple	82.68	86	89	87	88.4
LR- Tuned	83.11	86	89	87	88.4
LDA -Simple	83.33	86	89	88	88.9
LDA-Tuned	83.33	86	89	88	88.8
KNN -Simple	82.02	85	88	87	87.3
KNN- Tuned	82.90	84	91	88	88.9
Naïve Bayes - Simple	82.24	87	87	87	87.6
Naïve Bayes - Tuned	82.02	84	89	87	88.1
Bagging – Simple	82.02	86	88	87	88
Bagging- Tuned	79.61	86	83	84	85.6
Random Forest- Simple	83.11	85	90	88	89.5
Random Forest- Tuned	81.80	82	92	87	89.1
Ada Boost - Simple	82.02	86	87	87	87.8
Ada Boost - Tuned	80.92	83	89	86	88.4
Gradient Boost- Simple	83.33	85	91	88	89.9
Gradient Boost- Tuned	83.11	85	91	88	89.8

Table 11 Comparison of test data for all the Models

Comparison of AUC ROC curve on train data of all tuned Models

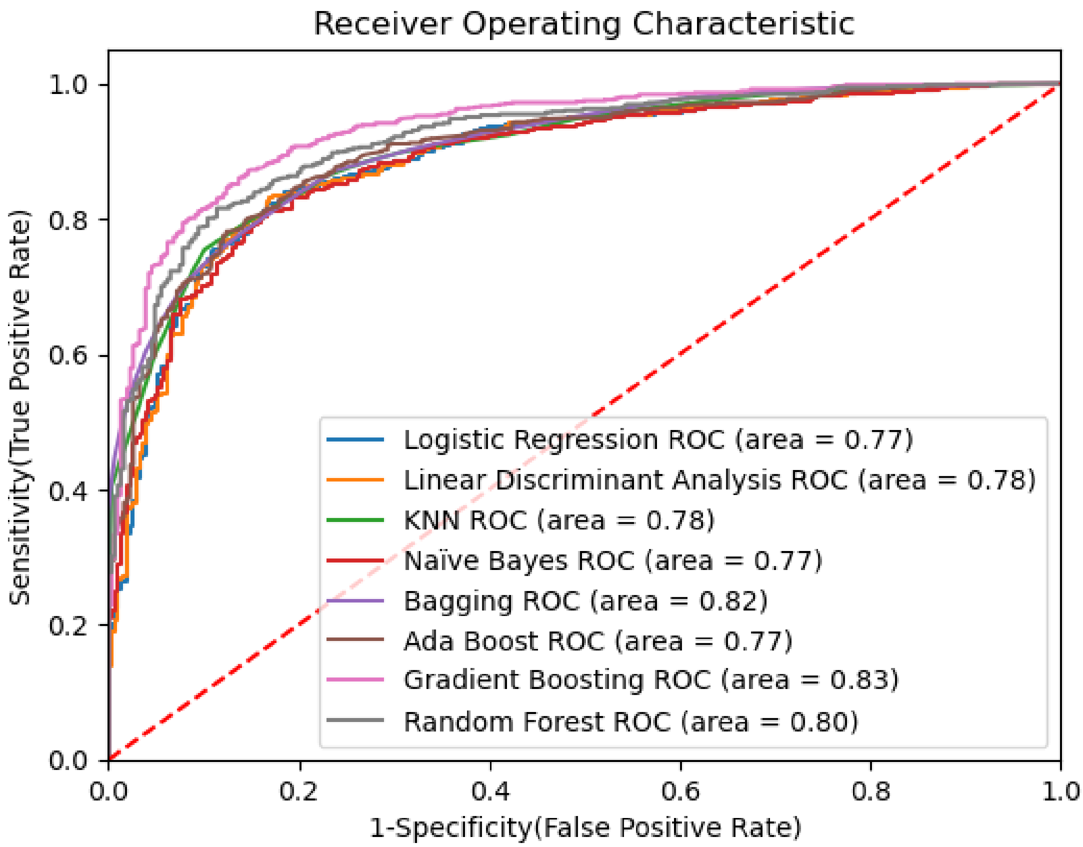


Fig.43 Comparison of AUC ROC curve on train data for all the Models

Comparison of AUC ROC curve on test data of all tuned Models

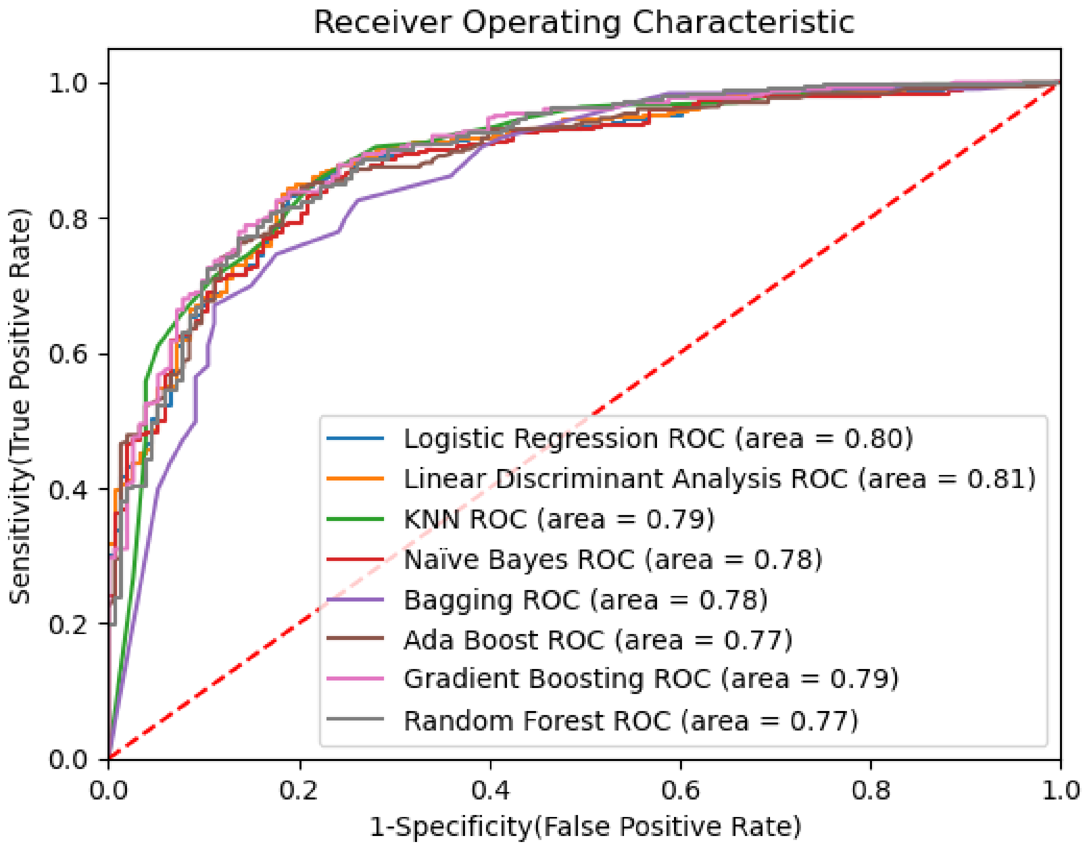


Fig.44 Comparison of AUC ROC curve on test data for all the Models

Conclusion

- There is no underfitting or overfitting in any of the tuned Models.
- The tuned gradient boost model performs the best with 89% accuracy in train data, indicating that it correctly predicts the outcome for a significant proportion of instances in the dataset.
- It also has a precision score of 91% and recall of 94% which is also the highest of all the models.
- The model successfully identifies 94% of all positive instances. This is crucial in situations where missing positive instances is costly.
- In this case, when the model predicts a positive result, it is correct 91% of the time. This is important in scenarios where false positives are costly or undesirable.
- **Gradient boost would be the best choice** as there is a balance among all the matrices like precision, recall and F1 score.

1.8 Based on these predictions, what are the insights?

Insights

- The Labour party has more than twice the number of votes compared to the Conservative party.
- The majority of respondents rated the national economic condition with scores of 3 and 4, with an average score of 3.245221.
- The household economic condition received predominantly scores of 3 and 4, with an average score of 3.137772.
- Blair garnered more votes than Hague, and the voter sentiment was significantly more favorable toward Blair than Hague.
- Blair's average score is 3.335531, while Hague's is 2.749506, indicating that Blair has a higher overall rating.
- About 30% of the population, when rated on a scale of 0 to 3, has zero knowledge about politics or parties.
- Despite giving a low score of 1 to a particular party, some voters still chose to vote for the same party, potentially indicating a lack of political knowledge among the electorate.
- Individuals with higher Eurosceptic sentiment tended to vote for the Conservative party, while those with lower Eurosceptic sentiment were more inclined to support the Labour party.
- Among those who scored 0 for political knowledge (454 people), 360 voted for the Labour party, and 94 voted for the Conservative party.
- All models exhibited strong performance on both the training and test datasets. Tuned models outperformed their regular counterparts.
- Overfitting is not observed in any model, except for the Random Forest and Bagging regular models.
- The Gradient Boosting model, post-tuning, emerged as the best-optimized model among the evaluated models.

Recommendations

- Effectively tuning hyperparameters is crucial in the model-building process. However, it's important to acknowledge the computational challenges associated with exploring a vast array of parameter combinations. Despite these limitations, experimenting with numerous parameter sets has the potential to yield superior results.

- Expanding the dataset is a valuable strategy for enhancing model training and, consequently, improving predictive capabilities. A larger dataset provides the models with more information, contributing to better overall performance.
- Implementing a sequential prediction function where models forecast outcomes in a sequence can enhance our understanding and offer valuable insights into the likelihood of various outcomes. This sequential approach can contribute to a more comprehensive comprehension of potential results.
- Leveraging the Gradient Boosting model is recommended, given its demonstrated superior performance after optimization. This model choice provides an efficient and effective solution for making accurate predictions without the additional complexity of scaling the data.

Problem 2:

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

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

(Hint: use. words (),. raw (),. sent() for extracting counts)The dataset is encoded using label encoding.

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

<u>Number of Characters</u>

```
Number of characters in Roosevelt file : 7571
Number of characters in Kennedy file : 7618
Number of characters in Nixon file : 9991
```

- President Franklin D. Roosevelt's speech has 7571characters (including spaces).
- President John F. Kennedy's speech has 7618 characters (including spaces)
- President Richard Nixon's speech has 9991 characters (including spaces)

<u>Number of Words</u>

	Speech	word_count
0	On each national day of inauguration since 178...	1323
1	Vice President Johnson, Mr. Speaker, Mr. Chief...	1364
2	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	1769

- There are 1323 words in President Franklin D. Roosevelt's speech.
- There are 1364 words in President John F. Kennedy's speech
- There are 1769 words in President Richard Nixon's speech

<u>Number of sentences</u>

	Text	sentences
0	On each national day of inauguration since 178...	67

	Text	sentences
0	Vice President Johnson, Mr. Speaker, Mr. Chief...	52

	Text	sentences
0	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	68

2.2 Remove all the stop words from all three speeches.

Before removing all the stop words from the speeches, we have changed all the letters to lower case and removed punctuations.

<u>Number of stop words</u>

	Speech	stopwords
0	On each national day of inauguration since 178...	632
1	Vice President Johnson, Mr. Speaker, Mr. Chief...	618
2	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	899

- Number of stop words in President Franklin Roosevelt’s speech are 632.
- Number of stop words in President John F. Kennedy’s speech are 618.
- Number of stop words in President Richard Nixon’s speech are 899.

	Name	Speech	word_count	stopwords	Processed_Speech	word_count_after_stopwords_removal
0	Roosevelt	On each national day of inauguration since 178...	1323	632	national day inauguration since 1789 people re...	624
1	Kennedy	Vice President Johnson, Mr. Speaker, Mr. Chief...	1364	618	vice president johnson mr speaker mr chief jus...	689
2	Nixon	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	1769	899	mr vice president mr speaker mr chief justice ...	819

Fig.45 Dataset after removal of stop words

<u>Dataset after removal of stop words</u>

Here, Processed Speech is the output of Speech after converting words into lower case and removal of all the punctuations.

After removal of stop words:

- President Franklin D. Roosevelt's speech has 624 words.
- President John F. Kennedy’s speech has 689 words.
- President Richard Nixon’s speech has 819 words.

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

<u>Top three words in President Franklin D. Roosevelt's speech</u>

```
Top 3 words in Roosevelt speech are:

nation    10
know      10
us         8
dtype: int64
```

<u>Top three words in President John F. Kennedy's speech</u>

```
Top 3 words in Kennedy speech are:

us        11
let       11
sides      8
dtype: int64
```

<u>Top three words in President Richard Nixon's speech</u>

```
Top 3 words in Nixon speech are:

us        26
new       15
peace     15
dtype: int64
```

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

<u>Word cloud of President Franklin D. Roosevelt's speech</u>

Fig.46 Word cloud of Roosevelt's speech

Word cloud of President John F. Kennedy’s speech

Fig.47 Word cloud of Kennedy’s speech

<u>Word cloud of President Richard Nixon's speech</u>

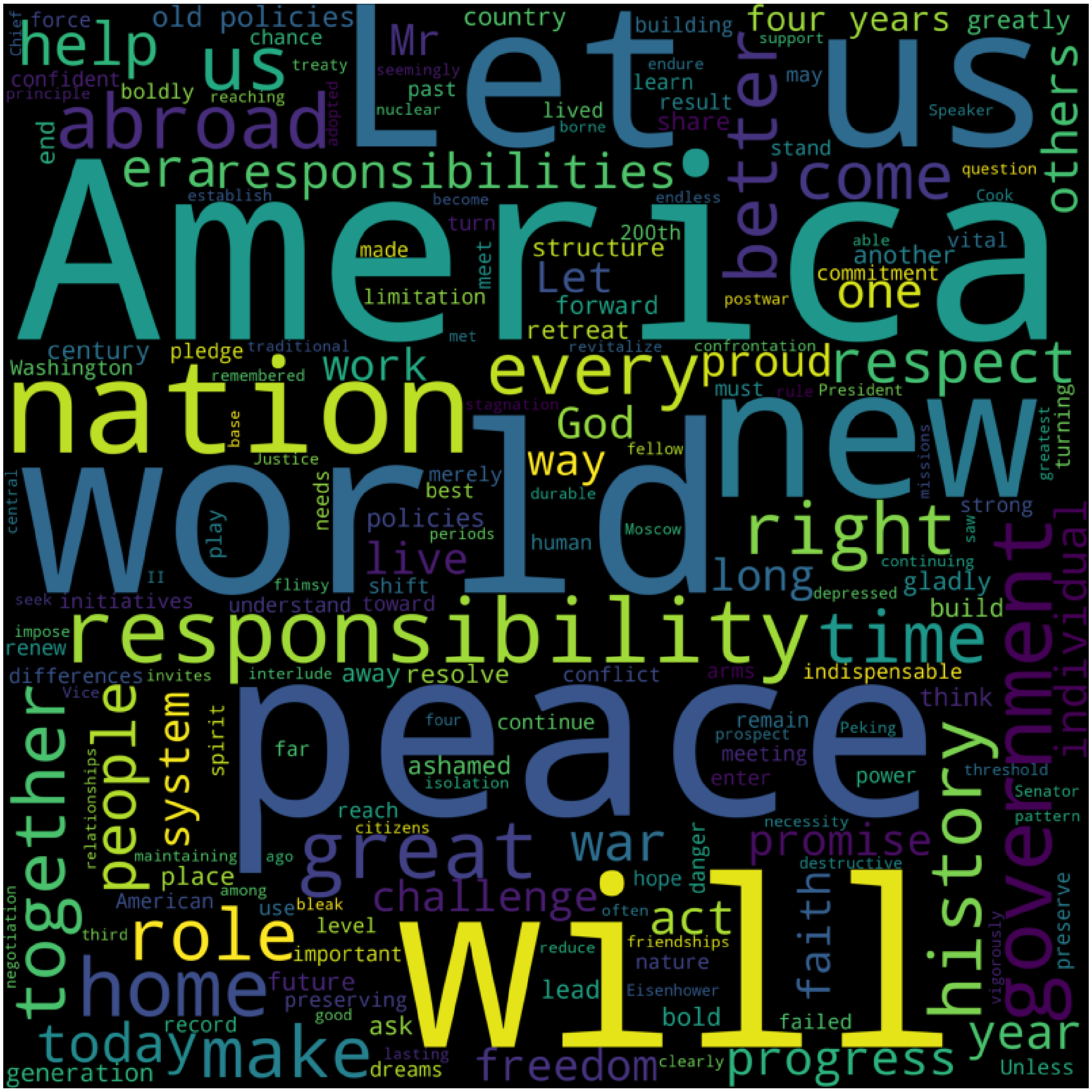


Fig.48 Word cloud of Nixon's speech