MACHINE LEARNING

Project Report

Submitted by,

Sindhu R Udupa.

BATCH: PGPDSBA.O. NOV22.B

Contents

List of Figures

List of Tables

Part -1

In this part, we are analyzing the election data procured by the news channel CNBE. The survey is conducted on 1525 voters with 9 variables. We are going 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.

The data dictionary of the data is as follows.

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.

1.1) 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 has been read. It has 1525 records and 10 columns. Among the 10 columns, one column is an id column. Among the remaining 9 columns, the column 'vote' is the dependent variable and the remaining are independent variables.

	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

Table 1: First five records of the data.

	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

Table 2: Last five records of the data.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 10 columns):
 #   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)
memory usage: 119.3+ KB
```

Table 3: Information of the data.

As we see from the above table, there are two columns of object data type and eight columns of the integer data type. The unnamed column is a unique identifier, we will drop it. The column ‘vote’ is our target variable. Among the remaining columns, the ‘age’ column is numerical in nature. The remaining columns, even though their data types are integers, are categorical in nature.

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
vote	1525	2	Labour	1063	NaN	NaN	NaN	NaN	NaN	NaN	NaN
age	1525.0	NaN	NaN	NaN	54.182295	15.711209	24.0	41.0	53.0	67.0	93.0
economic.cond.national	1525.0	NaN	NaN	NaN	3.245902	0.880969	1.0	3.0	3.0	4.0	5.0
economic.cond.household	1525.0	NaN	NaN	NaN	3.140328	0.929951	1.0	3.0	3.0	4.0	5.0
Blair	1525.0	NaN	NaN	NaN	3.334426	1.174824	1.0	2.0	4.0	4.0	5.0
Hague	1525.0	NaN	NaN	NaN	2.746885	1.230703	1.0	2.0	2.0	4.0	5.0
Europe	1525.0	NaN	NaN	NaN	6.728525	3.297538	1.0	4.0	6.0	10.0	11.0
political.knowledge	1525.0	NaN	NaN	NaN	1.542295	1.083315	0.0	0.0	2.0	2.0	3.0
gender	1525	2	female	812	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Table 4: Statistical description of the data.

From the table above, we see that 1063 voters are voting for labour party, which tells us that the data is imbalanced. The minimum and maximum age of the voters are 24 and 93 respectively. Blair is more popular than Hague since his mean assessment is more than that of Hague.

The null value check has been done. There are no null values present in the data set. Since the data came with a unique identifier column, we can say that data is not duplicated.

```
age                      0.144621
economic.cond.national  -0.240453
economic.cond.household -0.149552
Blair                   -0.535419
Hague                    0.152100
Europe                  -0.135947
political.knowledge     -0.426838
dtype: float64
```

Table 5: Skewness of the data.

We see that the absolute value of skewness of each column is approximately not more than 0.5. we can say the distribution is almost symmetrical. Some columns have positive skewness which implies right skewness and two of the columns have positive skewness, which implies the data is left skewed.

```
COLUMN NAME :  vote

Labour          1063
Conservative     462
Name: vote, dtype: int64
-------------------------

COLUMN NAME :  economic.cond.national

3    607
4    542
2    257
5     82
1     37
Name: economic.cond.national, dtype: int64
-------------------------

COLUMN NAME :  economic.cond.household

3    648
4    440
2    280
5     92
1     65
Name: economic.cond.household, dtype: int64
-------------------------

COLUMN NAME :  Blair

4    836
2    438
5    153
1     97
3      1
Name: Blair, dtype: int64
-------------------------

COLUMN NAME :  Hague

2    624
4    558
1    233
5     73
3     37
Name: Hague, dtype: int64
-------------------------

COLUMN NAME :  Europe

11    338
6     209
3     129
4     127
5     124
8     112
9     111
1     109
10    101
7      86
2      79
Name: Europe, dtype: int64
-------------------------

COLUMN NAME :  political.knowledge

2    782
0    455
3    250
1     38
Name: political.knowledge, dtype: int64
-------------------------

COLUMN NAME :  gender

female    812
male      713
Name: gender, dtype: int64
-------------------------
```

Table 6: Value count of all categories.

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.

The null value check, data types, shape of the dataset are already discussed earlier. We will move on to univariate analysis.

Univariate analysis:

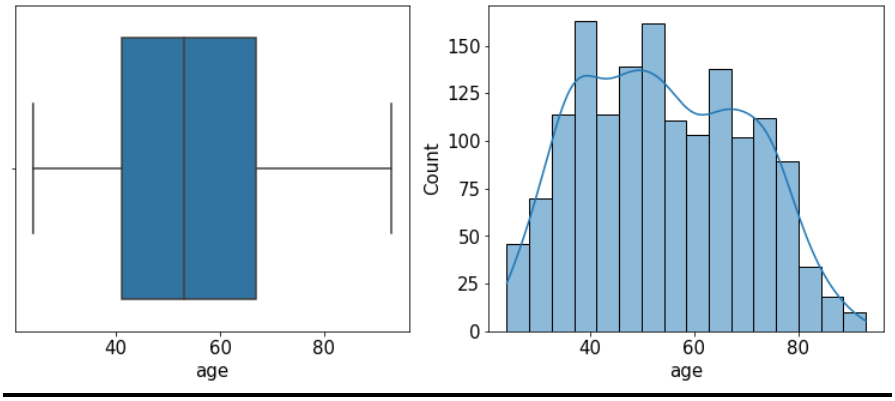

Fig 1: Boxplot and histograms of the column ‘age’.

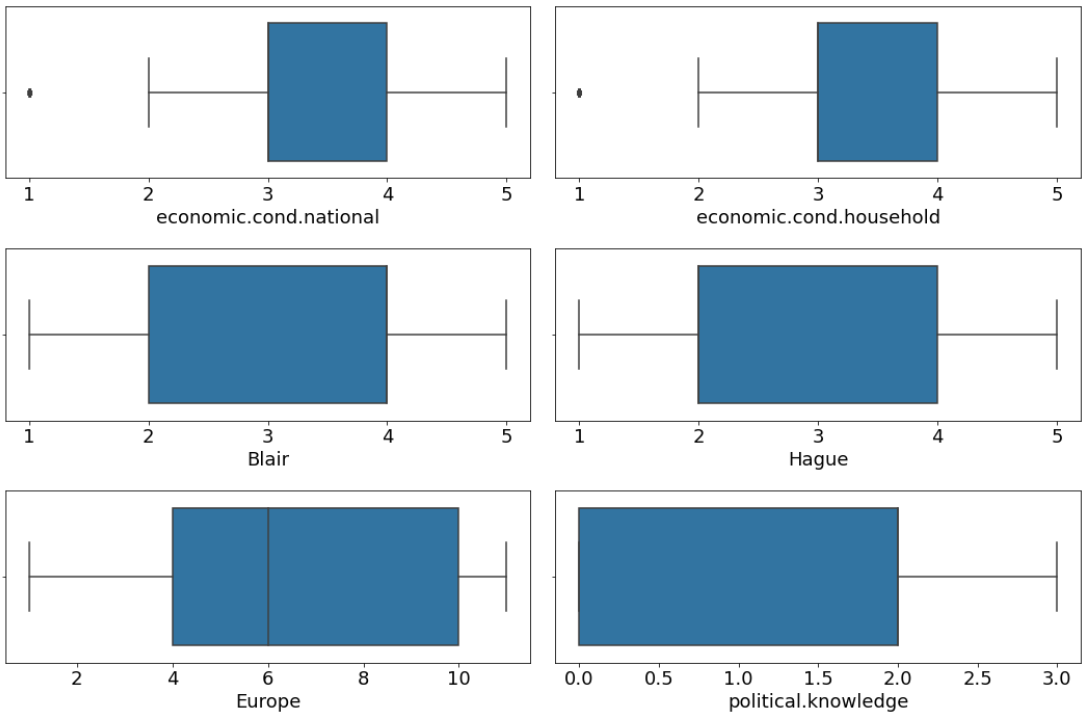

Fig 2: Boxplots of categorical variables.

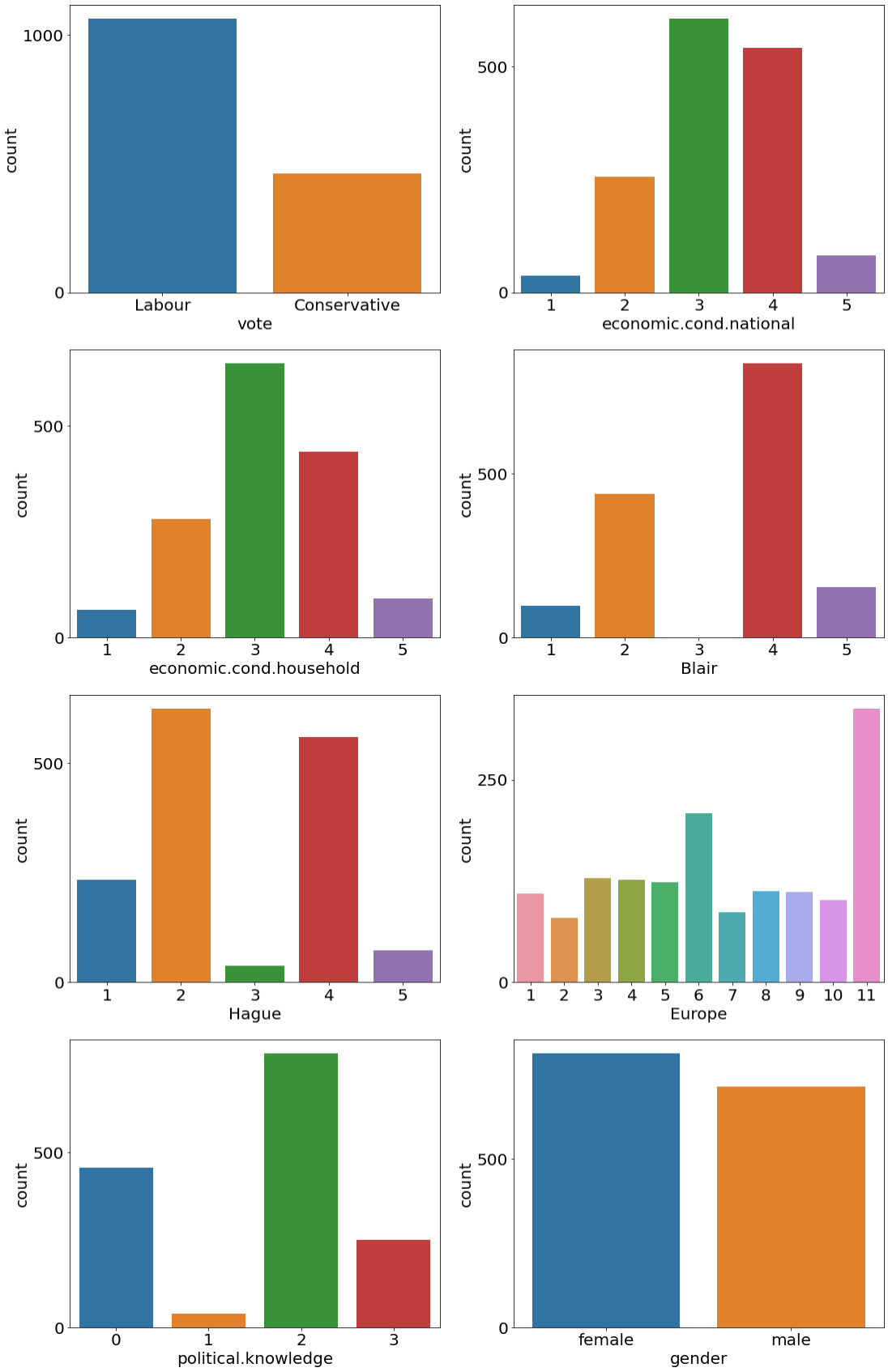

Fig 3: Count plots of categorical variables.

From fig. 1,

- The age column has no outliers. And the distribution is slightly right skewed.

From fig. 2,

- We can see outliers in the columns ‘national economic condition’ and ‘household economic condition’. But we are not going to treat these outliers since data is not erroneous.
- The outliers present in those columns tells us that there are only very few households and very few nations whose economic conditions are low.

From fig. 3,

- We see that data is imbalanced since we see more points for labour party than for conservatives.
- The number of countries who have moderate economic conditions are more.
- There are only a few countries who are very poor or very rich.
- Majority of the people reflect Eurosceptic sentiment, where as a few are neutral in this matter.

Bivariate Analysis:

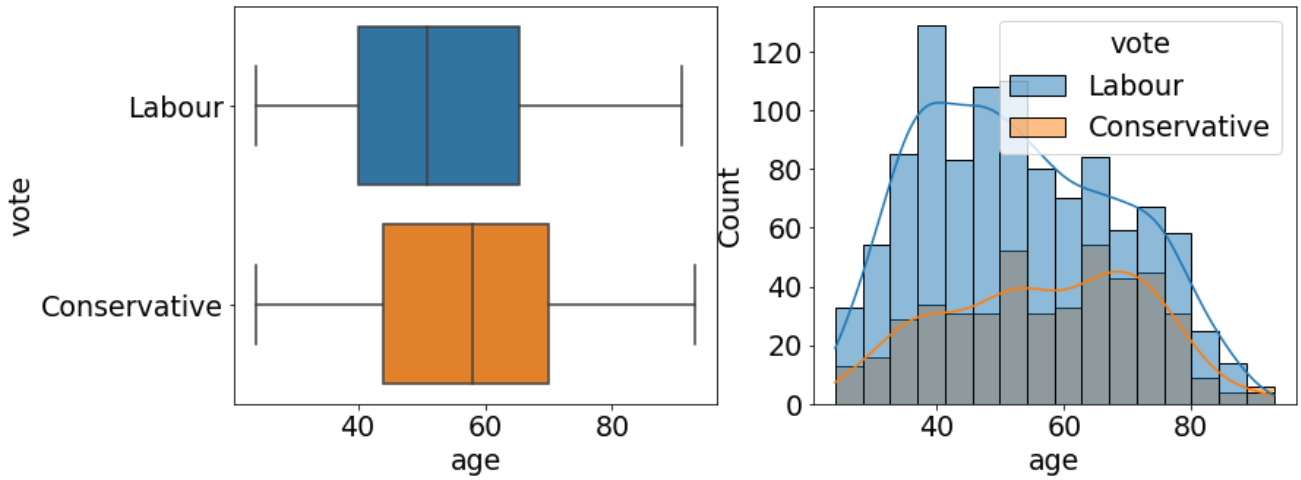

Fig 4: Age v/s Vote.

From the box plot above, we see that young people are more likely to vote for Labour party, whereas the older people prefer the conservative party.

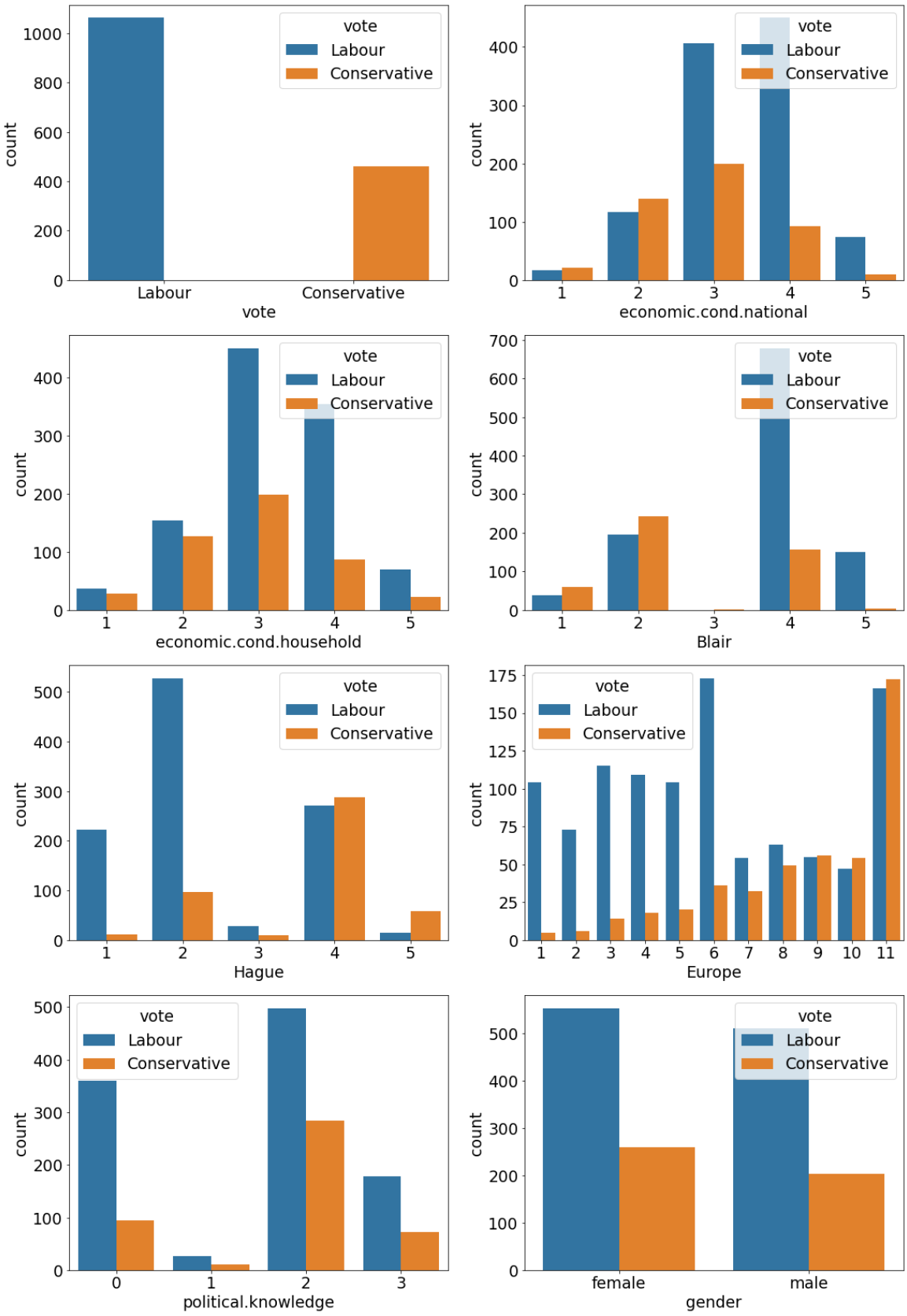

Fig 5: Categorical variables v/s Target variable.

From the above figure,

- The nations in which economic conditions are moderate, they're more likely to support Labour party.
- Similarly, the households in which the economic conditions are moderate, they are also more likely to support labour party.
- We see that the voters favoring labour party giving more assessment rating to their candidate Blair.
- People who are against the Eurosceptic sentiment are more likely to be the supporters of the labour party.
- Among the people who express Eurosceptic sentiment, there is a good share of voters from both the parties.
- There are only a few people who have very high knowledge of politics. Among them labour party supporters are more.

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.

In the data, the columns gender and vote have string values. Gender column is one hot encoded using get dummies function of pandas. The target column vote is custom encoded. Conservative party is given a value of 0 and Labour party is given a value of 1.

From table 4, we see the standard deviation of the age column is 15.71 and the standard deviation of all other columns are between 0-3, we don't see the necessity to scale the data for all the models. However, since the KNN model is Euclidean distance based, we shall scale the data for that model.

The data has been split into independent variables (X) and dependent variable (y). They are further split into train and test data in the ratio 70:30 using the train_test_split function from sklearn. The train set contains 1067 records and the test set contains 458 records.

We have used stratify parameter on the vote column to make sure the same ratio of Labour and Conservative parties is present in both train and test data.

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
1493	34	3	1	4	2	6	2	0
1431	42	3	4	4	4	3	2	0
235	36	4	4	4	2	7	2	0
1078	64	4	3	2	1	4	2	1
735	37	4	4	4	2	2	3	1

Table 7: First five records (independent variables) of train set.

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
1317	32	3	4	4	4	9	0	1
229	68	2	3	5	2	8	0	0
609	84	4	2	4	1	4	3	0
24	60	3	2	4	4	2	2	0
1479	79	4	4	5	1	3	3	1

Table 8: First five records (independent variables) of test set.

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)

Logistic Regression

We have implemented Logistic Regression using the functions from sklearn library. We have not tuned any hyper parameters at the moment.

Accuracy of the model on Training data: 0.8294.
Accuracy of the model on Testing data: 0.8493.

When we see the accuracy scores of the Logistic Regression model, we do not see any over-fitting issue. However, since the data is imbalanced, the accuracy scores are not reliable. When we check the f1 scores, we find that f1 score of class 0 is 0.73 where as that of class 1 is 0.90 for the test data.

Linear Discriminant Analysis

We have implemented Linear discriminant analysis using the sklearn library. We have not tuned any hyper parameters at the moment.

Accuracy of the model on Training data: 0.8256
Accuracy of the model on Testing data: 0.8449

The accuracy scores do not seem to be over-fitting. The LDA model is working fine in terms of precision, recall and f1 score. We do not see any over fitting issues. The f1 score of class 0 is 0.73 and of class 1 is 0.89 in test data.

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)

K Nearest Neighbors

Before we apply the KNN algorithm, we will scale the data using z-score from scipy library.

We are only scaling the numerical variable 'age'. We are not variables such as 'economic conditions of house holds', 'Blair', 'Hague', etc. because they are not numerical but ordinal variables.

The KNN model has been implemented successfully using a function from the sklearn library, without tuning any hyper parameters.

Accuracy of the model on Training data: 0.8622.
Accuracy of the model on Testing data: 0.8362.

Even though the accuracy of test set is less than that of the train set, we don't consider this to be overfitting because the drop in test score is less than 10% of the train score.

We can say that KNN model is doing a good job in classifying the votes. The f1 score for class 0 is 0.72 and for class 1 is 0.88 in the test data.

Naïve Bayes

Naïve Bayes model is implemented using the sklean library. No hyper parameters are tuned for this model. The accuracy scores are as follows.

Accuracy of the model on Training data: 0.8219
Accuracy of the model on Testing data: 0.8471

As we clearly see there is no over fitting in this model. The f1 score for class 0 is 0.74 and for class 1 is 0.89 for test set. The accuracy scores for test data is more than that of train data.

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

Tuned Logistic regression model:

The hyper parameters used to tune this model are as follows:

a. 'solver' : ('newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga')
b. 'penalty' : ('l1', 'l2', 'elasticnet', 'none')
c. 'C' : (100, 10, 1.0, 0.1, 0.01)}

The best combination of the hyper parameters are as follows:

'C' = 0.01, 'penalty' = 'l2', 'solver' = 'newton-cg'

Another Logistic regression model has been built using these hyper parameters. Its accuracy scores are as follows.

Accuracy of the model on Training data: 0.8266
Accuracy of the model on Testing data: 0.8449.

By looking at the numbers, we don't see any great improvement in the logistic regression model after tuning it.

Tuned Linear Discriminant Model:

The hyper parameters used to tune this model are as follows:

a. 'solver' : ('svd', 'lsqr', 'eigen')

The best combination of the hyper parameter is as follows:

Solver= 'svd', which was the default value in the model. Hence we conclude that our base LDA model is the best LDA model without any hyper parameter tuning.

Tuned KNN Model:

The hyper parameters used to tune this model are as follows:

a. 'n_neighbors': 1,3,5,7,9,11,13,15,17,19

b. 'algorithm': ('auto', 'ball_tree', 'kd_tree' , 'brute')

c. 'weights' : ('uniform', 'distance')

The best combination of the hyper parameters are as follows:

'algorithm'= 'brute', 'n_neighbors'= 13, 'weights' ='uniform'

Another KNN model has been built using these hyper parameters. Its accuracy scores are as follows.

Accuracy of the model on Training data: 0.8388.
Accuracy of the model on Testing data: 0.8427.

We see some improvement in the KNN model after tuning the hyper parameters. The untuned KNN model was slightly over fitting (even though, not according to the industry standards) That issue is resolved after the tuning process.

Tuned Random Forest:

The hyper parameters used to tune this model are as follows:

a. 'n_estimators' : [50,100,200],
b. 'criterion' : ["gini", "entropy"],
c. 'max_depth':[2,4,5],
d. 'min_samples_leaf':[10,15,20],
e. 'min_samples_split': [30,45,60]

The best combination of the hyper parameter is as follows:

'criterion'= 'entropy', 'max_depth'= 5, 'min_samples_leaf'= 10, 'min_samples_split'= 30, 'n_estimators'= 100.

Accuracy of the model on Training data: 0.8500.
Accuracy of the model on Testing data: 0.8384.

The overfit, untuned random forest model was giving an accuracy score of 0.9990 for train data, where as the accuracy level dropped to 0.8493 for test data. This problem of over fitting has been addressed by tuning the model.

We can see that tuned random forest model is not an over fit. But its performing comparatively poorly when we consider the recall of class 0. We will see if this improves when we give this model to bagging.

Bagging:

We have implemented the bagging ensemble technique successfully using the sklearn library. We have used the tuned random forest model as the base estimator for bagging.

Accuracy of the model on Training data: 0.8397
Accuracy of the model on Testing data: 0.8493.
The f1 scores for the class 0 is 0.72 and that of class 1 is 0.90.

We see there is no over fitting in this model.

Ada Boosting:

The ada boost model is implemented using the sklearn library. We have not tuned any hyper parameters here.

Accuracy of the model on Training data: 0.8444
Accuracy of the model on Testing data: 0.8296

Even though we see the accuracy score for train data is more than the accuracy score for te st data, we can not consider this to be a case of over fitting by industry standards.

Gradient Boosting:

The gradient boost model is implemented using the sklearn library. We have not tuned any hyper parameters here.

Accuracy of the model on Training data: 0.8875
Accuracy of the model on Testing data: 0.8384

We see a slight over fitting in gradient boosting. The accuracy scores are dropping by 5%.

Tuned Bagging:

Since it is asked to keep random forest as the base estimator of the bagging, we have not tuned this parameter. The other parameter used for tuning is 'n_estimators'. The default value is 10. When tuned, the best value for this parameter has come out to be 30. We have created another model with this value. The accuracy scores of the model are as follows:

Accuracy of the model on Training data: 0.8406.
Accuracy of the model on Testing data: 0.8427.
There is no overfitting in this model.

Tuned Ada Boost:

The parameters used for tuning are 'n_estimators' and algorithm. The default value of n_estimator is 50. When tuned, the best value for this parameter has come out to be the same. The default algorithm was SAMME.R, but it has come out to be SAMME when tuned. We have created another model with these parameter values. The accuracy scores of the model are as follows:

Accuracy of the model on Training data: 0.8369.
Accuracy of the model on Testing data: 0.8427.
There is no overfitting in this model.

Tuned Gradient Boost:

The parameters used for tuning are 'n_estimators', criterion and loss. The default value of n_estimator is 100. When tuned, the best value for this parameter has come out to be the same. The default criterion was friedman_mse, and it has come out to be the same. The tuned value for 'loss' is 'exponential'. We have created another model with these parameter values. The accuracy scores of the model are as follows:

Accuracy of the model on Training data: 0.8847.
Accuracy of the model on Testing data: 0.8427.
We see a slight overfitting in the model.

Feature Importance:

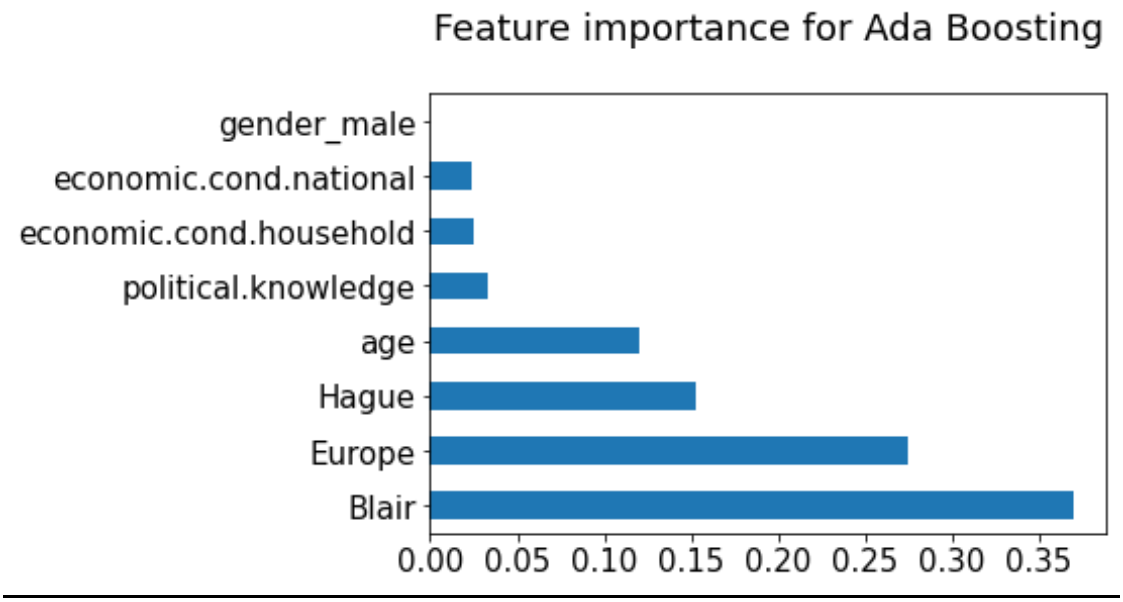

Fig. 6: Feature importance of Ada Boosting

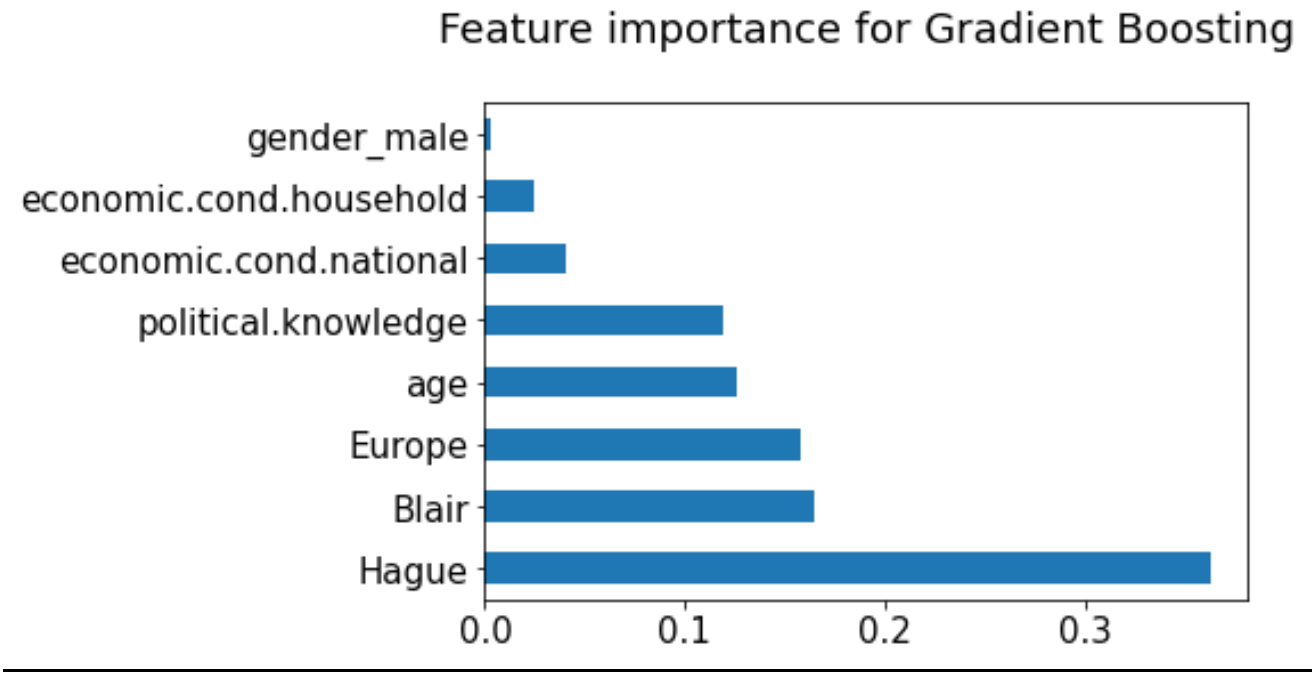

Fig. 7: Feature importance of Gradient Boosting

We see that the column Blair has the highest weightage when building the Ada boost model and the column Hague has the highest weightage when building gradient boost model.

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)

Performance Metrics on each model

Classification Reposts: The classification report contains the accuracy scores, class level metrics such as precision, recall, f1 score. We shall look at the classification report of each model for train and test data.

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.75	0.65	0.70	323
1	0.86	0.91	0.88	744
accuracy			0.83	1067
macro avg	0.80	0.78	0.79	1067
weighted avg	0.83	0.83	0.83	1067

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.80	0.68	0.73	139
1	0.87	0.92	0.90	319
accuracy			0.85	458
macro avg	0.83	0.80	0.81	458
weighted avg	0.85	0.85	0.85	458

Table 9: Classification Report of train and test data-Logistic Regression.

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.73	0.67	0.70	323
1	0.86	0.90	0.88	744
accuracy			0.83	1067
macro avg	0.80	0.78	0.79	1067
weighted avg	0.82	0.83	0.82	1067

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.77	0.70	0.73	139
1	0.87	0.91	0.89	319
accuracy			0.84	458
macro avg	0.82	0.80	0.81	458
weighted avg	0.84	0.84	0.84	458

Table 10: Classification Report of train and test data-LDA.

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.79	0.74	0.77	323
1	0.89	0.91	0.90	744
accuracy			0.86	1067
macro avg	0.84	0.83	0.83	1067
weighted avg	0.86	0.86	0.86	1067

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.75	0.68	0.72	139
1	0.87	0.90	0.88	319
accuracy			0.84	458
macro avg	0.81	0.79	0.80	458
weighted avg	0.83	0.84	0.83	458

Table 11: Classification Report of train and test data-KNN.

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.71	0.69	0.70	323
1	0.87	0.88	0.87	744
accuracy			0.82	1067
macro avg	0.79	0.78	0.79	1067
weighted avg	0.82	0.82	0.82	1067

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.76	0.73	0.74	139
1	0.88	0.90	0.89	319
accuracy			0.85	458
macro avg	0.82	0.81	0.82	458
weighted avg	0.85	0.85	0.85	458

Table 12: Classification Report of train and test data-Naïve Bayes.

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.77	0.60	0.68	323
1	0.84	0.92	0.88	744
accuracy			0.83	1067
macro avg	0.81	0.76	0.78	1067
weighted avg	0.82	0.83	0.82	1067

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.80	0.65	0.72	139
1	0.86	0.93	0.89	319
accuracy			0.84	458
macro avg	0.83	0.79	0.81	458
weighted avg	0.84	0.84	0.84	458

Table 13: Classification Report of train and test data-Tuned Logistic Regression.

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.75	0.70	0.72	323
1	0.87	0.90	0.89	744
accuracy			0.84	1067
macro avg	0.81	0.80	0.80	1067
weighted avg	0.84	0.84	0.84	1067

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.77	0.69	0.73	139
1	0.87	0.91	0.89	319
accuracy			0.84	458
macro avg	0.82	0.80	0.81	458
weighted avg	0.84	0.84	0.84	458

Table 14: Classification Report of train and test data-Tuned KNN Model.

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.84	0.63	0.72	323
1	0.85	0.95	0.90	744
accuracy			0.85	1067
macro avg	0.84	0.79	0.81	1067
weighted avg	0.85	0.85	0.84	1067

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.83	0.59	0.69	139
1	0.84	0.95	0.89	319
accuracy			0.84	458
macro avg	0.83	0.77	0.79	458
weighted avg	0.84	0.84	0.83	458

Table 15: Classification Report of train and test data-Tuned Random Forest.

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.82	0.60	0.70	323
1	0.85	0.94	0.89	744
accuracy			0.84	1067
macro avg	0.83	0.77	0.79	1067
weighted avg	0.84	0.84	0.83	1067

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.84	0.63	0.72	139
1	0.85	0.95	0.90	319
accuracy			0.85	458
macro avg	0.84	0.79	0.81	458
weighted avg	0.85	0.85	0.84	458

Table 16: Classification Report of train and test data-Bagging

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.83	0.60	0.70	323
1	0.84	0.94	0.89	744
accuracy			0.84	1067
macro avg	0.84	0.77	0.79	1067
weighted avg	0.84	0.84	0.83	1067

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.84	0.60	0.70	139
1	0.84	0.95	0.89	319
accuracy			0.84	458
macro avg	0.84	0.77	0.80	458
weighted avg	0.84	0.84	0.83	458

Table 17: Classification Report of train and test data-Tuned Bagging

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.76	0.70	0.73	323
1	0.88	0.91	0.89	744
accuracy			0.84	1067
macro avg	0.82	0.80	0.81	1067
weighted avg	0.84	0.84	0.84	1067

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.74	0.68	0.71	139
1	0.86	0.90	0.88	319
accuracy			0.83	458
macro avg	0.80	0.79	0.79	458
weighted avg	0.83	0.83	0.83	458

Table 18: Classification Report of train and test data-Ada Boost

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.75	0.69	0.72	323
1	0.87	0.90	0.88	744
accuracy			0.84	1067
macro avg	0.81	0.80	0.80	1067
weighted avg	0.83	0.84	0.84	1067

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.76	0.70	0.73	139
1	0.87	0.91	0.89	319
accuracy			0.84	458
macro avg	0.82	0.80	0.81	458
weighted avg	0.84	0.84	0.84	458

Table 19: Classification Report of train and test data-Tuned Ada Boost

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.85	0.76	0.80	323
1	0.90	0.94	0.92	744
accuracy			0.89	1067
macro avg	0.88	0.85	0.86	1067
weighted avg	0.89	0.89	0.89	1067

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.77	0.66	0.71	139
1	0.86	0.92	0.89	319
accuracy			0.84	458
macro avg	0.82	0.79	0.80	458
weighted avg	0.83	0.84	0.83	458

Table 20: Classification Report of train and test data-Gradient Boost

Classification Report on Train Data

	precision	recall	f1-score	support
0	0.84	0.77	0.80	323
1	0.90	0.94	0.92	744
accuracy			0.88	1067
macro avg	0.87	0.85	0.86	1067
weighted avg	0.88	0.88	0.88	1067

Classification Report on Test Data

	precision	recall	f1-score	support
0	0.78	0.68	0.72	139
1	0.87	0.92	0.89	319
accuracy			0.84	458
macro avg	0.82	0.80	0.81	458
weighted avg	0.84	0.84	0.84	458

Table 21: Classification Report of train and test data-Tuned Gradient Boost

Confusion Matrix: The confusion matrix tells us that how many points are correctly classified and how many points are incorrectly classified. The entries on

- First row and first column: gives the number of actual class0 and predicted class0.
- First row and second column: gives the number of actual class0 and predicted class1.
- Second row and first column: gives the number of actual class1 and predicted class0.
- Second row and Second column: gives the number of actual class1 and predicted class1.

```
Confusion matrix on train data
[[210 113]
 [ 69 675]]

Confusion matrix on test data
[[ 94  45]
 [ 24 295]]
```

Table 22: Confusion Matrix -Logistic Regression

```
Confusion matrix on train data
[[215 108]
 [ 78 666]]

Confusion matrix on test data
[[ 97  42]
 [ 29 290]]
```

Table 23: Confusion Matrix -LDA

```
Confusion matrix on train data
[[240  83]
 [ 64 680]]

Confusion matrix on test data
[[ 95  44]
 [ 31 288]]
```

Table 24: Confusion Matrix -KNN

```
Confusion matrix on train data
[[223 100]
 [ 90 654]]

Confusion matrix on test data
[[101  38]
 [ 32 287]]
```

Table 25: Confusion Matrix -Naïve Bayes.

```
Confusion matrix on train data
[[195 128]
 [ 57 687]]

Confusion matrix on test data
[[ 91  48]
 [ 23 296]]
```

Table 26: Confusion Matrix -Tuned Logistic Regression

```
Confusion matrix on train data
[[225  98]
 [ 74 670]]

Confusion matrix on test data
[[ 96  43]
 [ 29 290]]
```

Table 27: Confusion Matrix -Tuned KNN

```
Confusion matrix on train data
[[203 120]
 [ 40 704]]

Confusion matrix on test data
[[ 82  57]
 [ 17 302]]
```

Table 28: Confusion Matrix -Tuned Random forest

```
Confusion matrix on train data
[[195 128]
 [ 43 701]]

Confusion matrix on test data
[[ 87  52]
 [ 17 302]]
```

Table 29: Confusion Matrix -Bagging

```
Confusion matrix on train data
[[194 129]
 [ 41 703]]

Confusion matrix on test data
[[ 83  56]
 [ 16 303]]
```

Table 30: Confusion Matrix -Tuned Bagging

```
Confusion matrix on train data
[[227  96]
 [ 70 674]]

Confusion matrix on test data
[[ 94  45]
 [ 33 286]]
```

Table 31: Confusion Matrix -Ada Boost

```
Confusion matrix on train data
[[224  99]
 [ 75 669]]

Confusion matrix on test data
[[ 97  42]
 [ 30 289]]
```

Table 32: Confusion Matrix -Tuned Ada Boost

```
Confusion matrix on train data
[[246  77]
 [ 43 701]]

Confusion matrix on test data
[[ 92  47]
 [ 27 292]]
```

Table 33: Confusion Matrix -Gradient Boost

```
Confusion matrix on train data
[[248  75]
 [ 48 696]]

Confusion matrix on test data
[[ 94  45]
 [ 27 292]]
```

Table 34: Confusion Matrix -Tuned Gradient Boost

ROC Curve:

ROC stands for Receiver Operating Characteristic curve. It is a probability curve. AUC stands for Area under the ROC curve. AUC score represents the degree of separability. Higher the AUC score, better the model is at predicting the classes correctly. We will see the ROC curves for different models. AUC scores are given in the summary table.

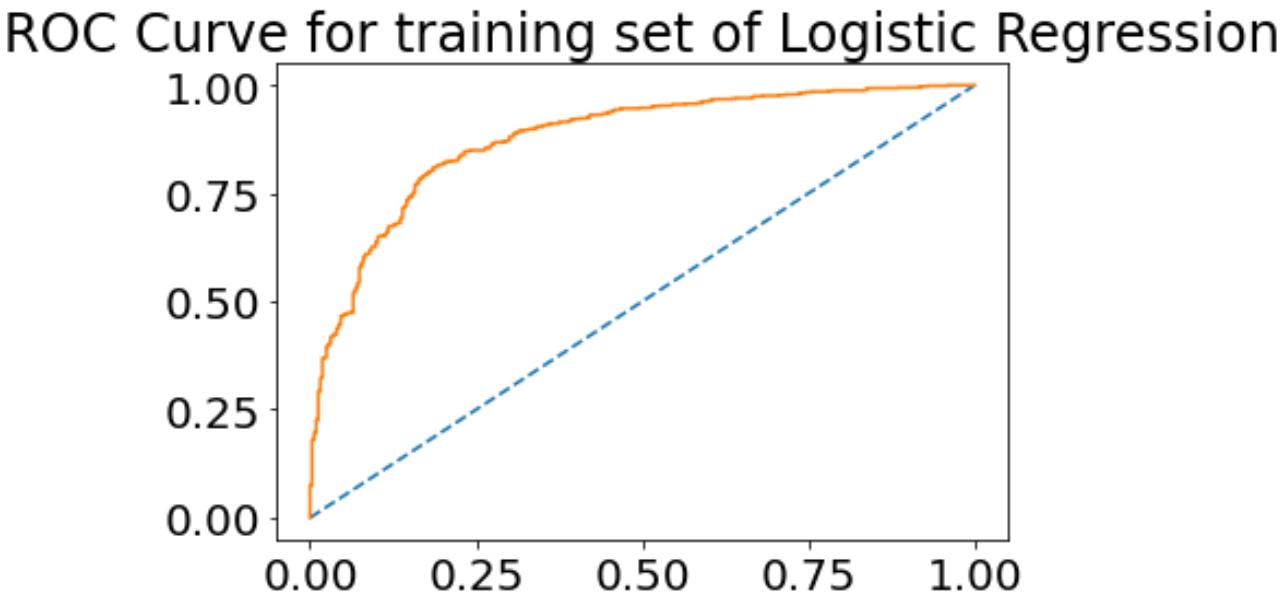

Fig. 8: ROC Curve – Logistic Regression – Train

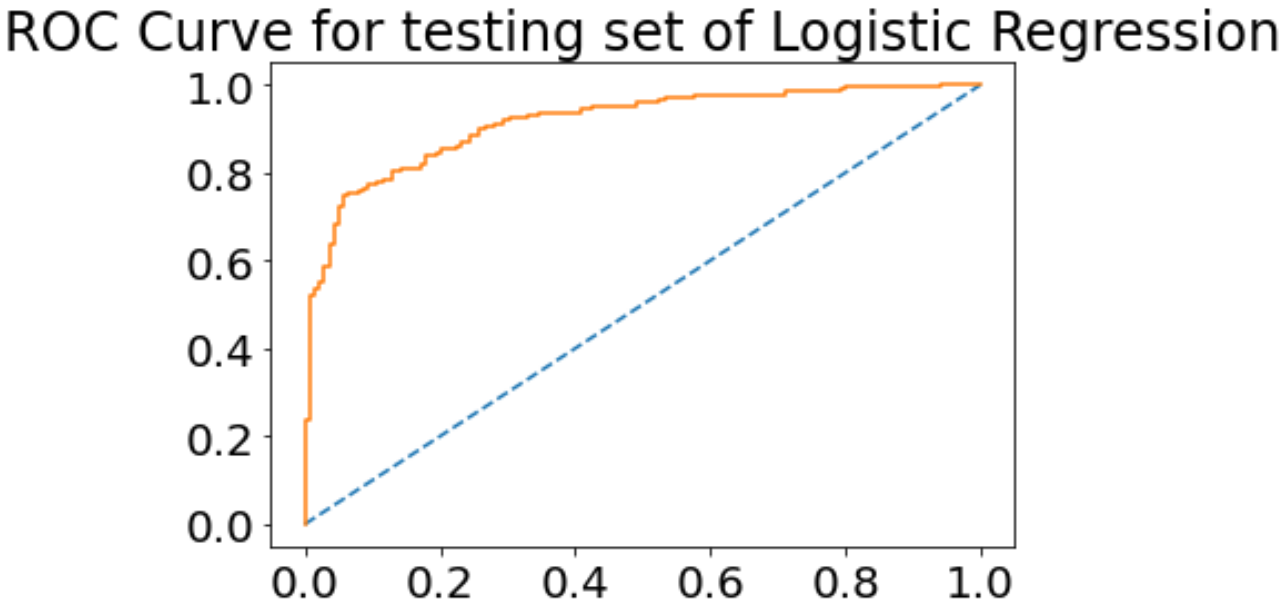

Fig. 9: ROC Curve – Logistic Regression - Test

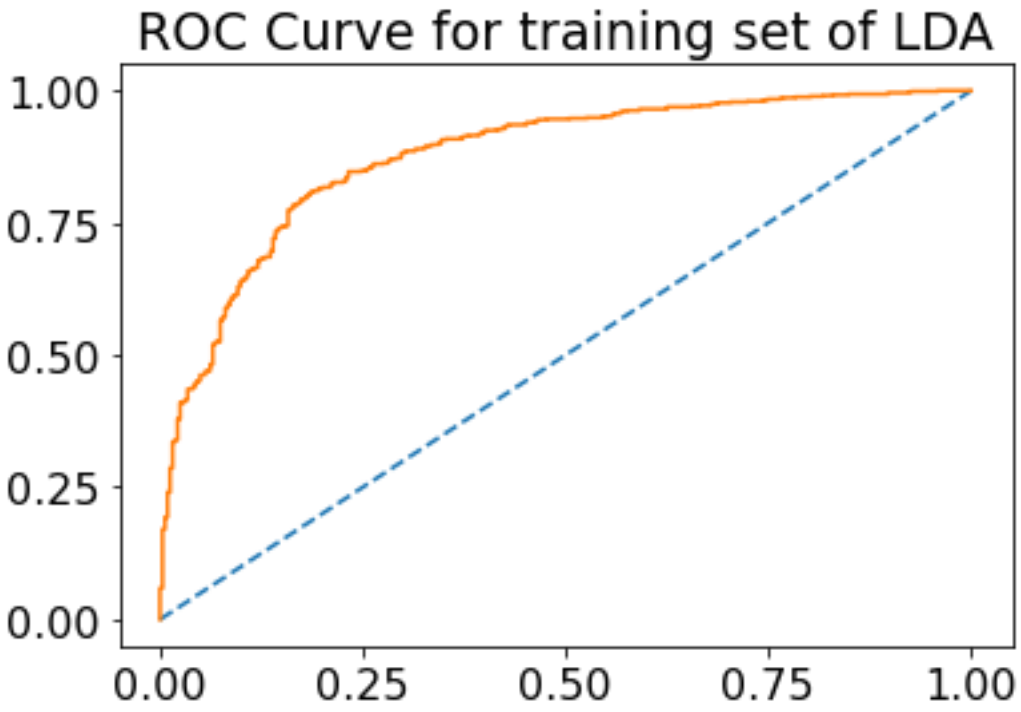

Fig. 10: ROC Curve – LDA - Train

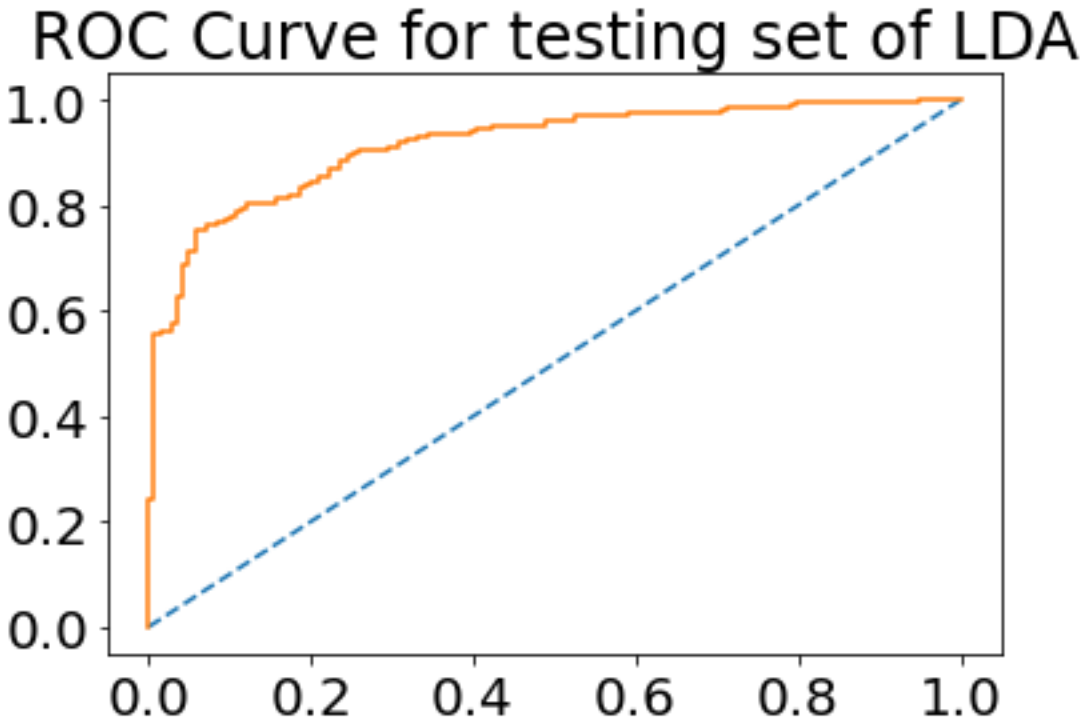

Fig. 11: ROC Curve – LDA – Test

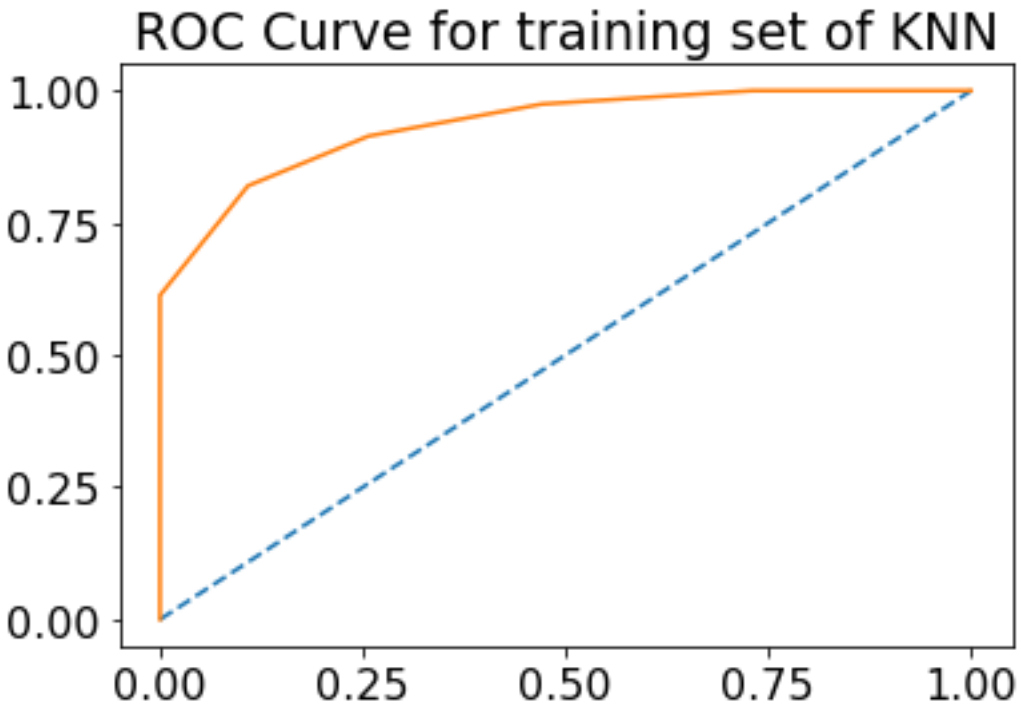

Fig. 12: ROC Curve – KNN - Train

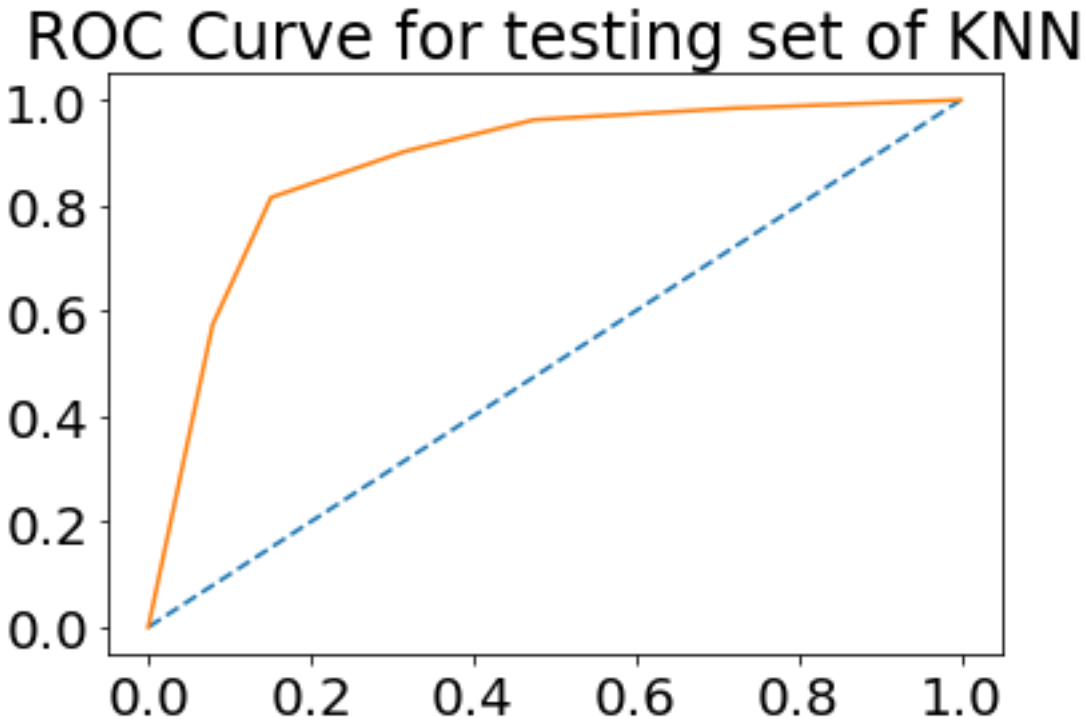

Fig. 13: ROC Curve – KNN - Test

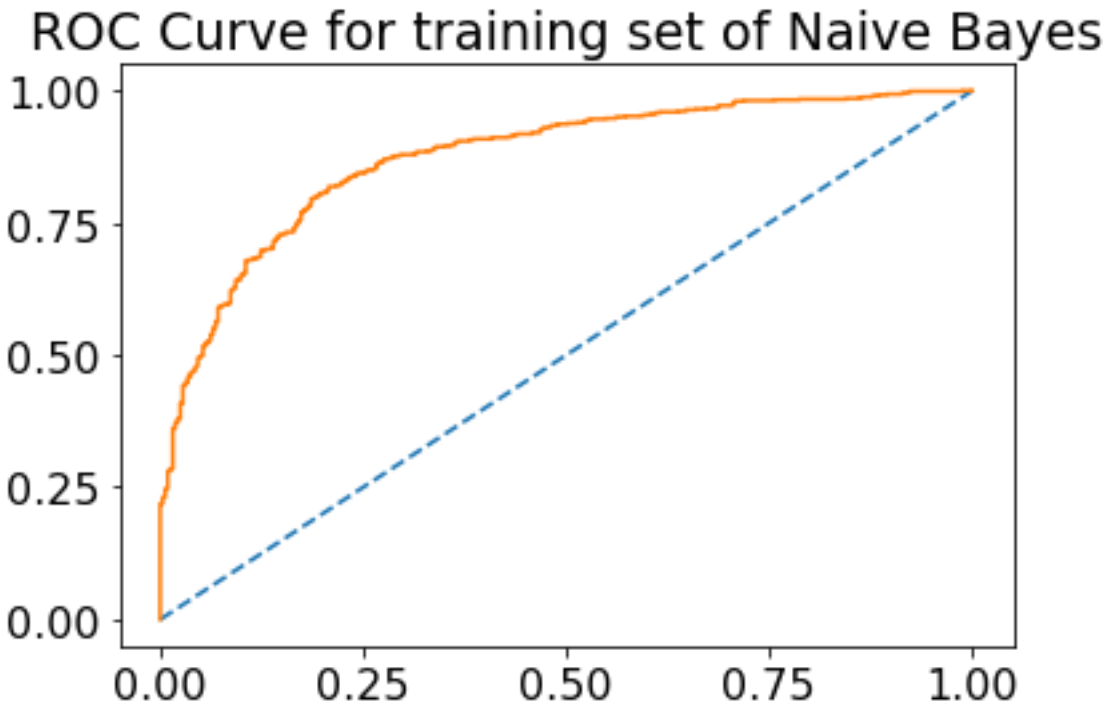

Fig. 14: ROC Curve – Naïve Bayes – Train

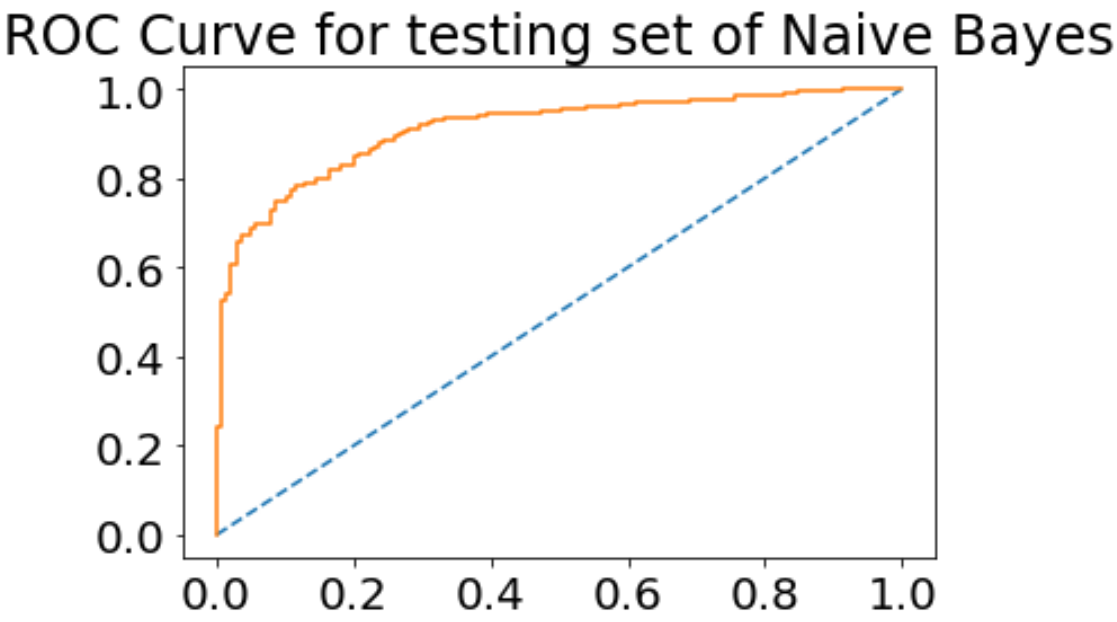

Fig. 15: ROC Curve – Naïve Bayes - Test

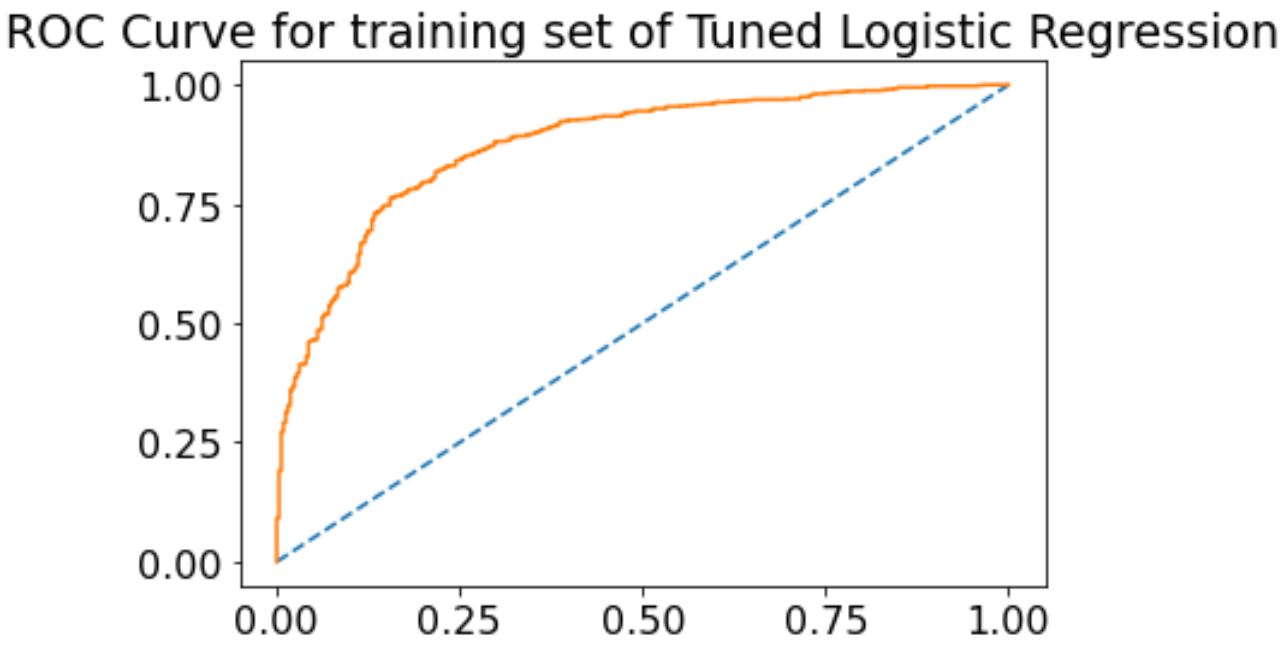

Fig. 16: ROC Curve – Tuned Logistic Regression – Train

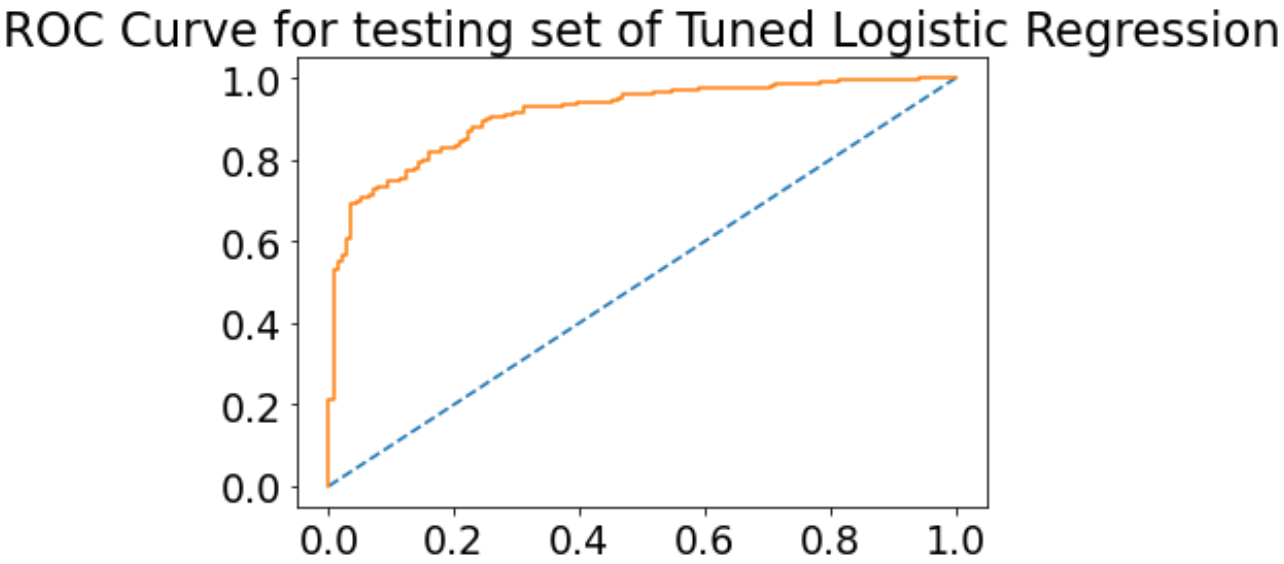

Fig. 17: ROC Curve – Tuned Logistic Regression - Test

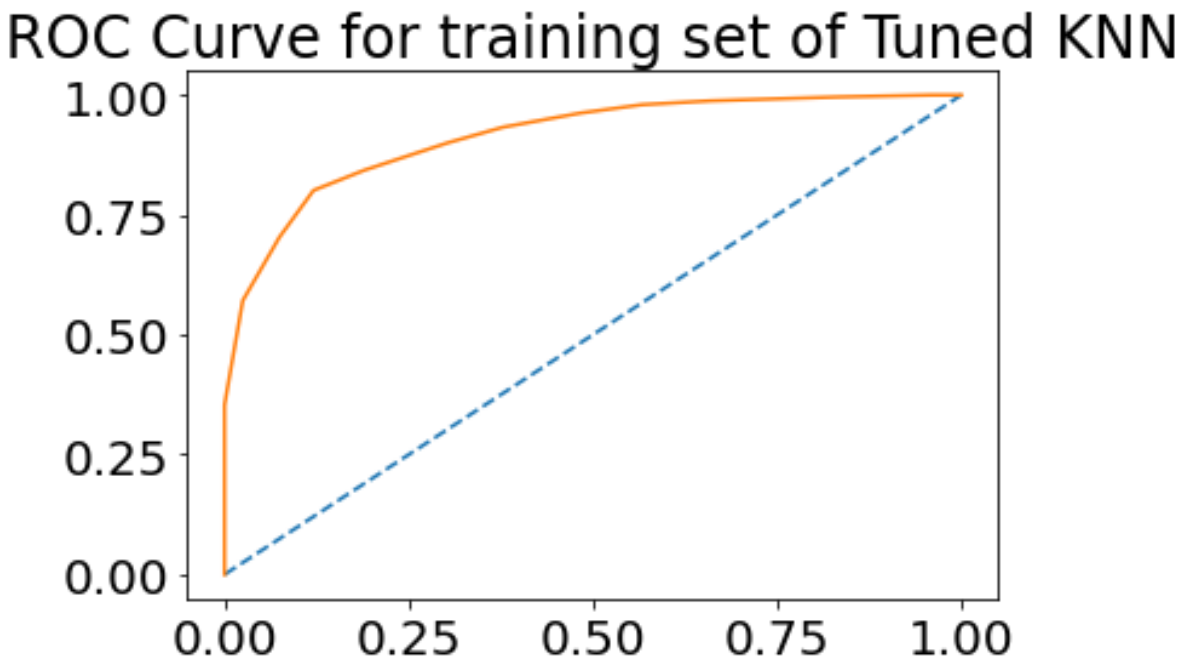

Fig. 18: ROC Curve – Tuned KNN – Train

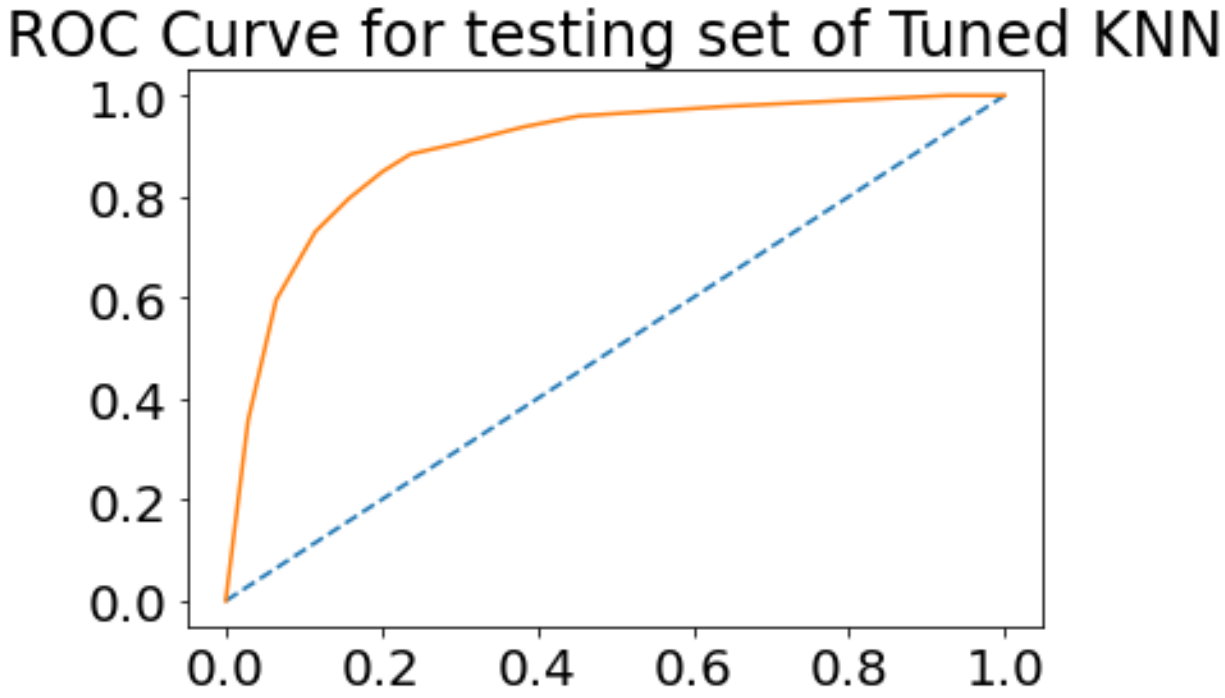

Fig. 19: ROC Curve – Tuned KNN - Test

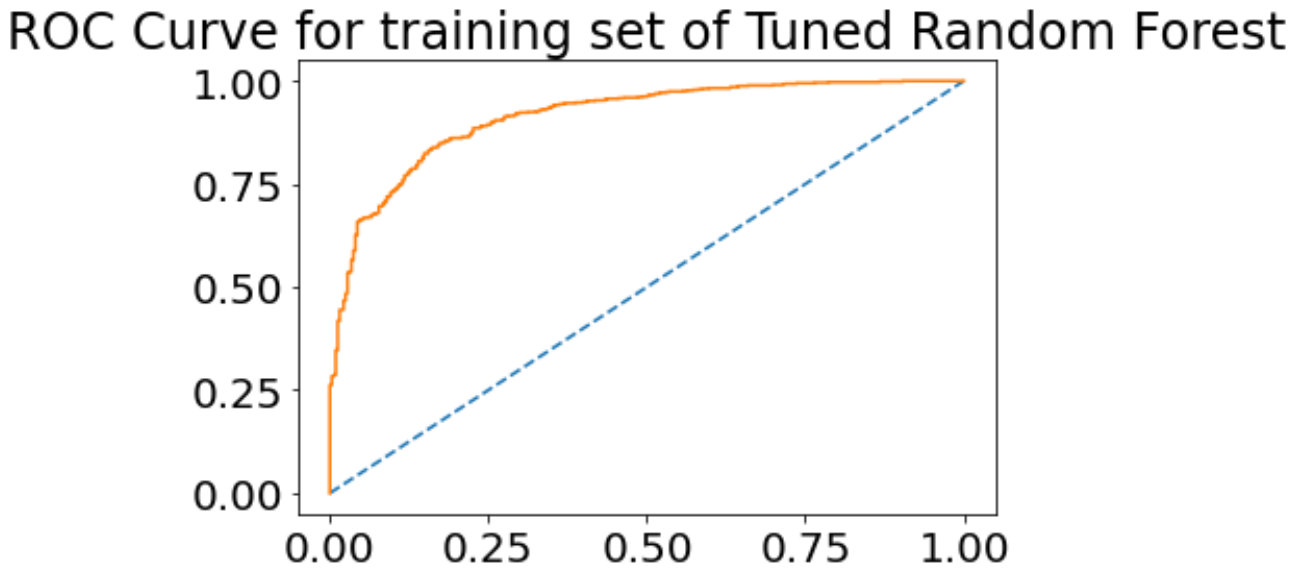

Fig. 20: ROC Curve – Tuned Random Forest – Train

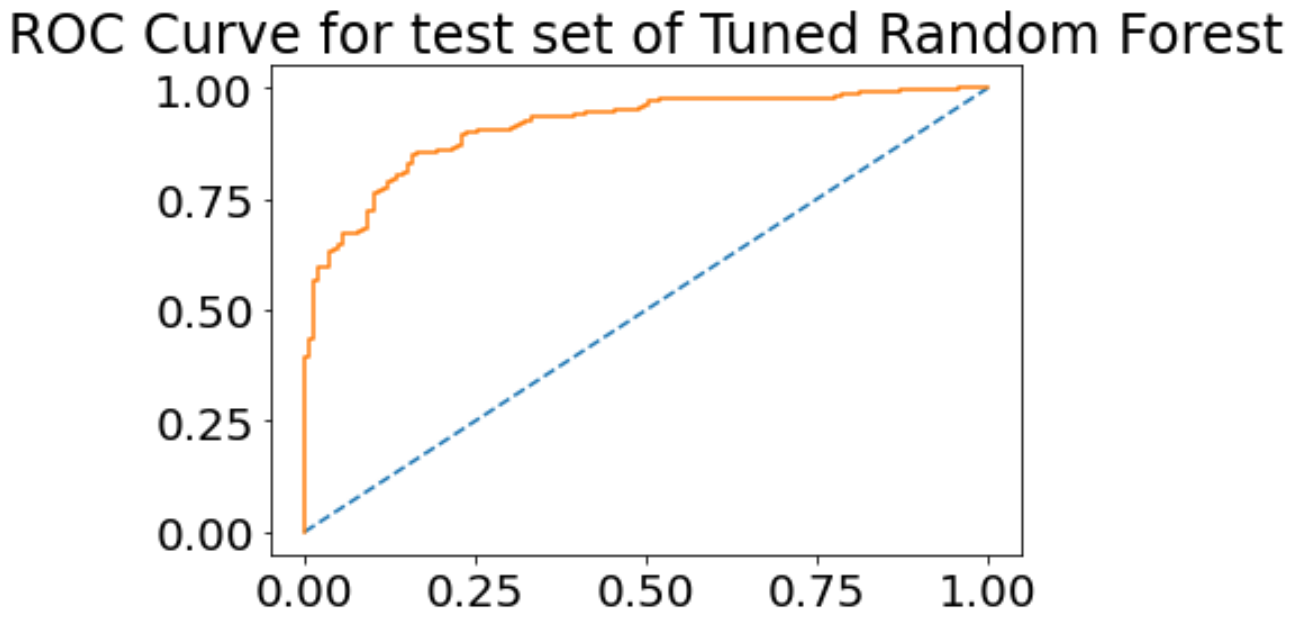

Fig. 21: ROC Curve – Tuned Random Forest - Test

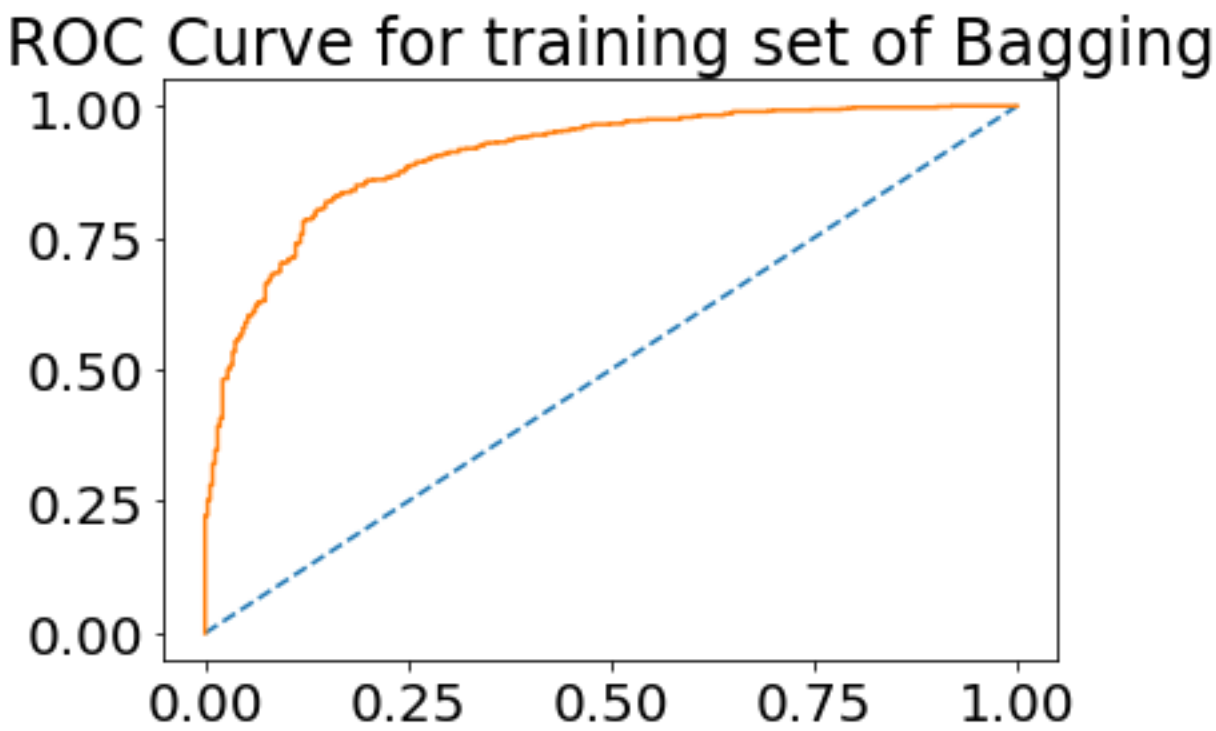

Fig. 22: ROC Curve – Bagging – Train

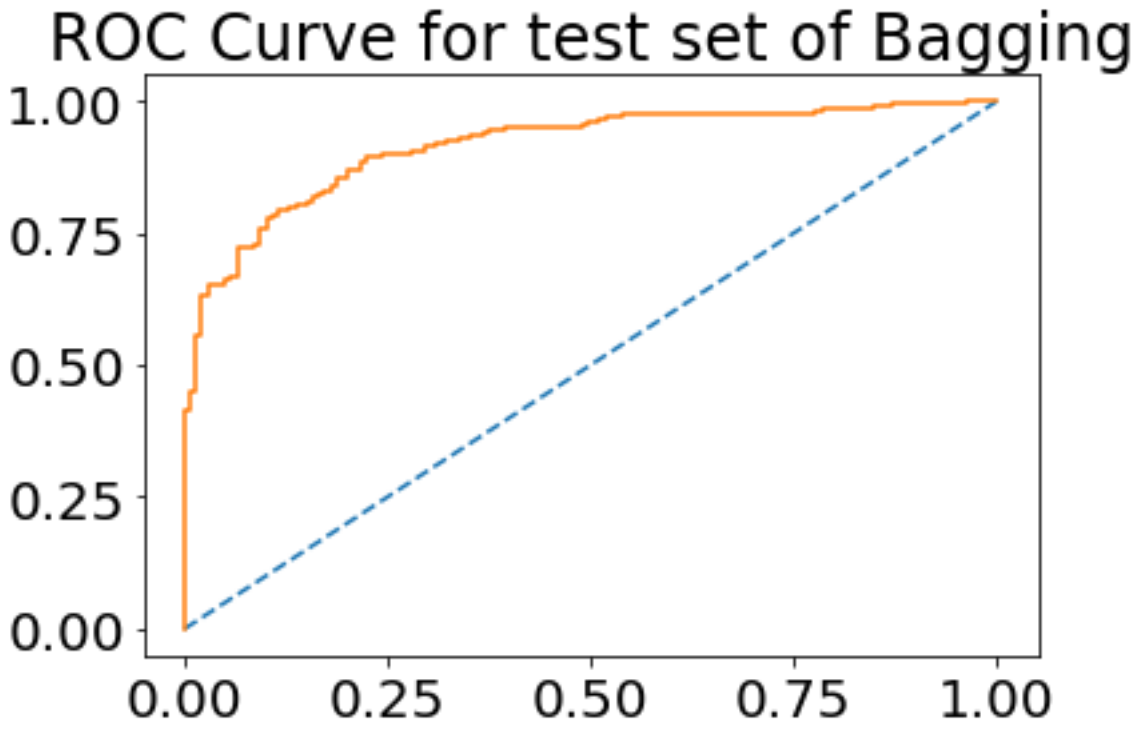

Fig. 23: ROC Curve – Bagging - Test

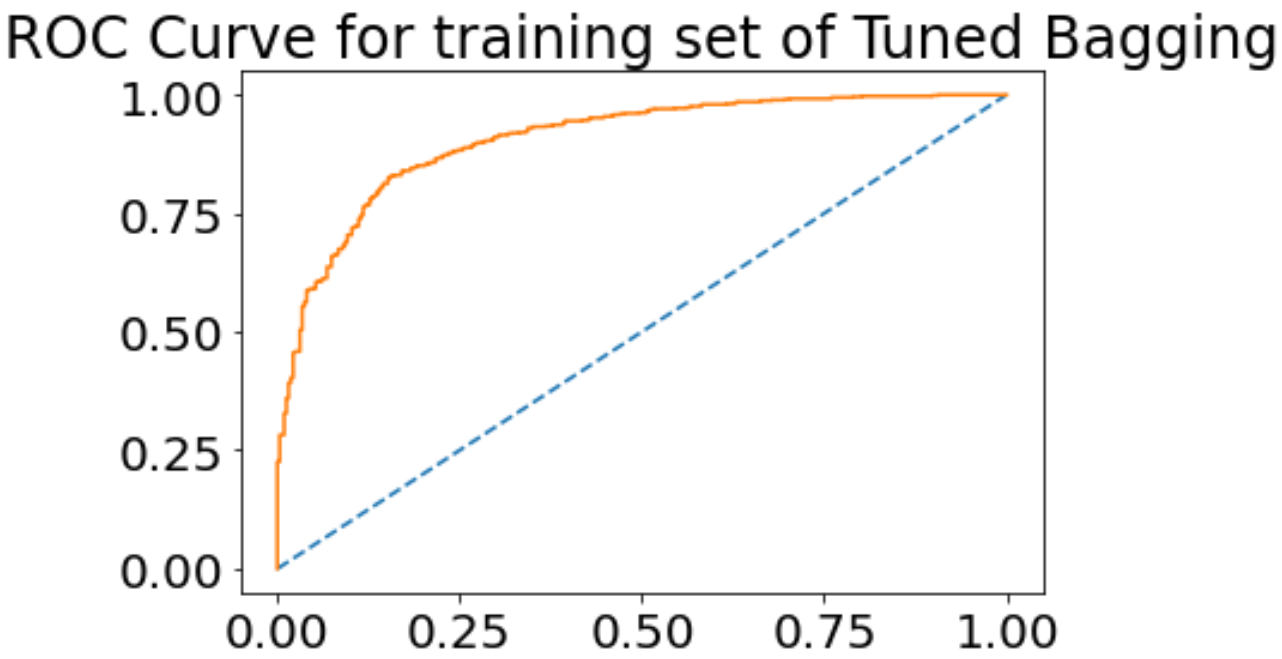

Fig. 24: ROC Curve – Tuned Bagging – Train

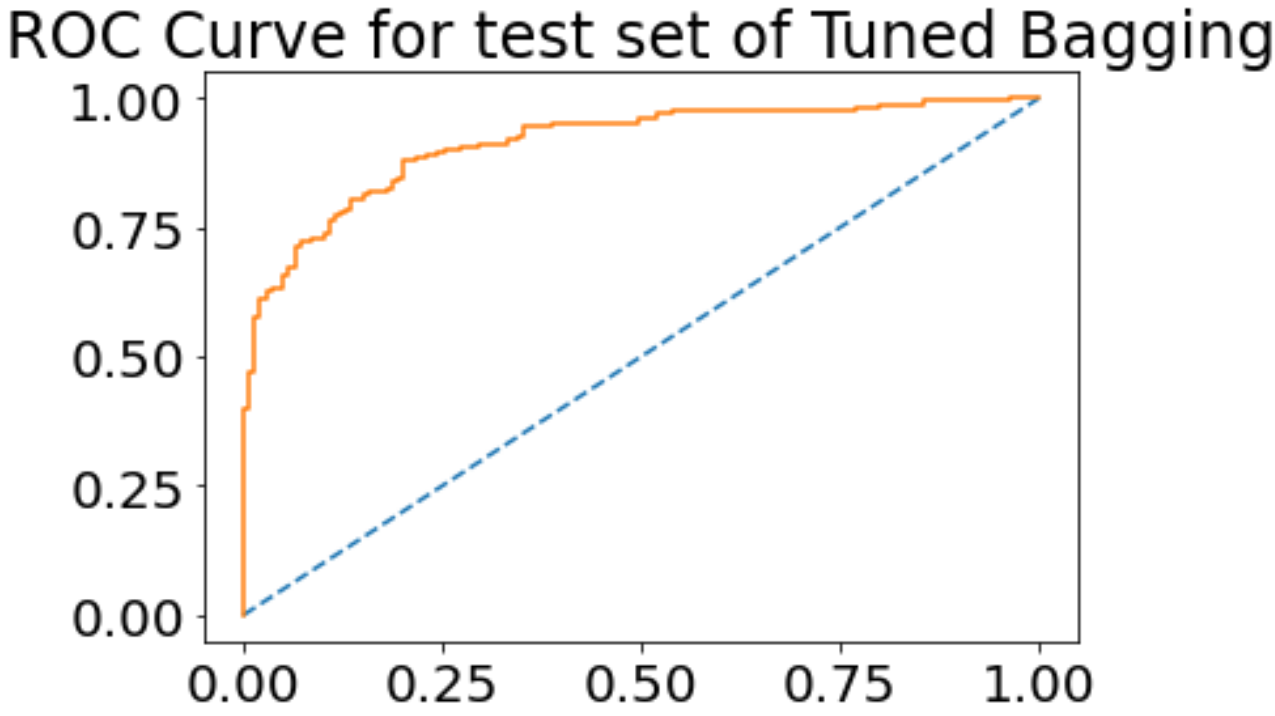

Fig. 25: ROC Curve – Tuned Bagging - Test

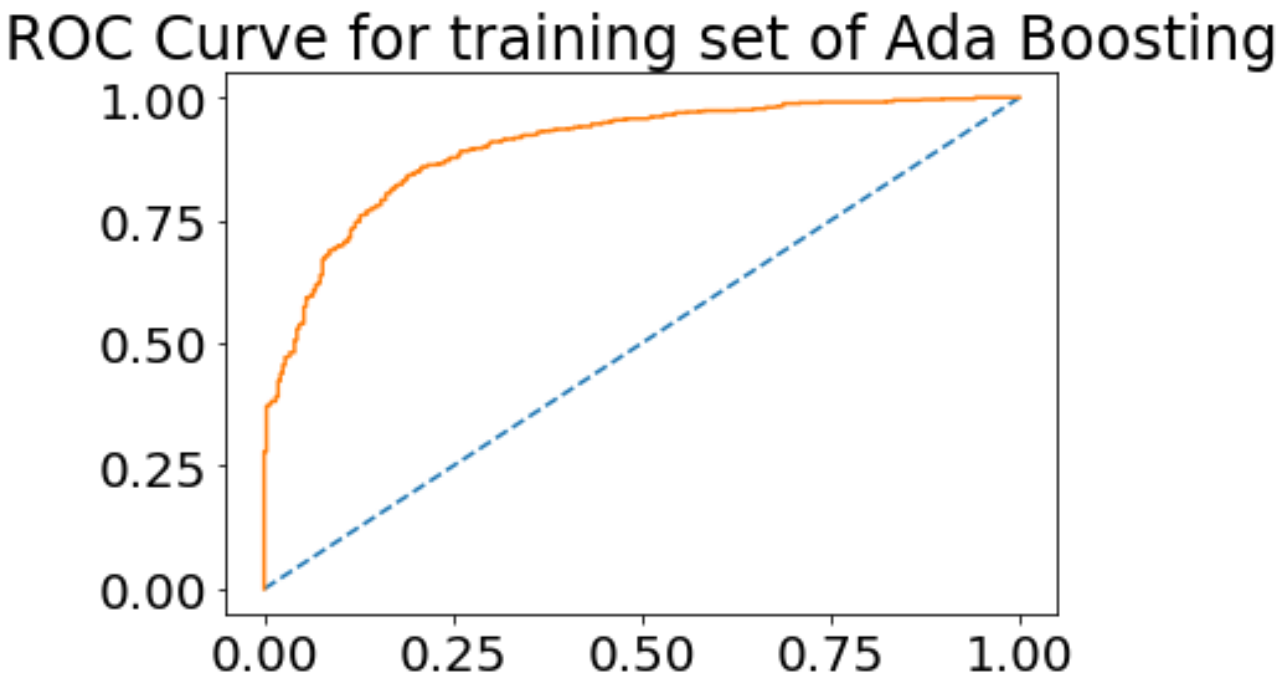

Fig. 26: ROC Curve – Ada Boost – Train

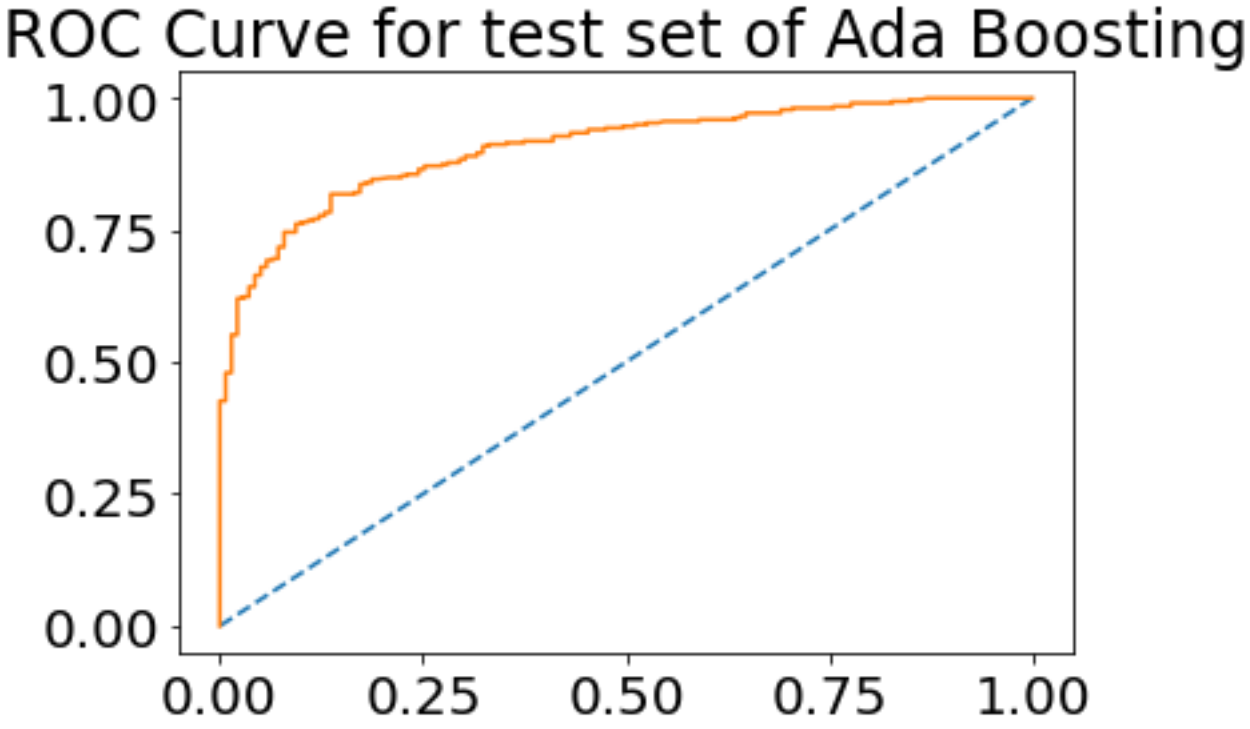

Fig. 27: ROC Curve – Ada Boost - Test

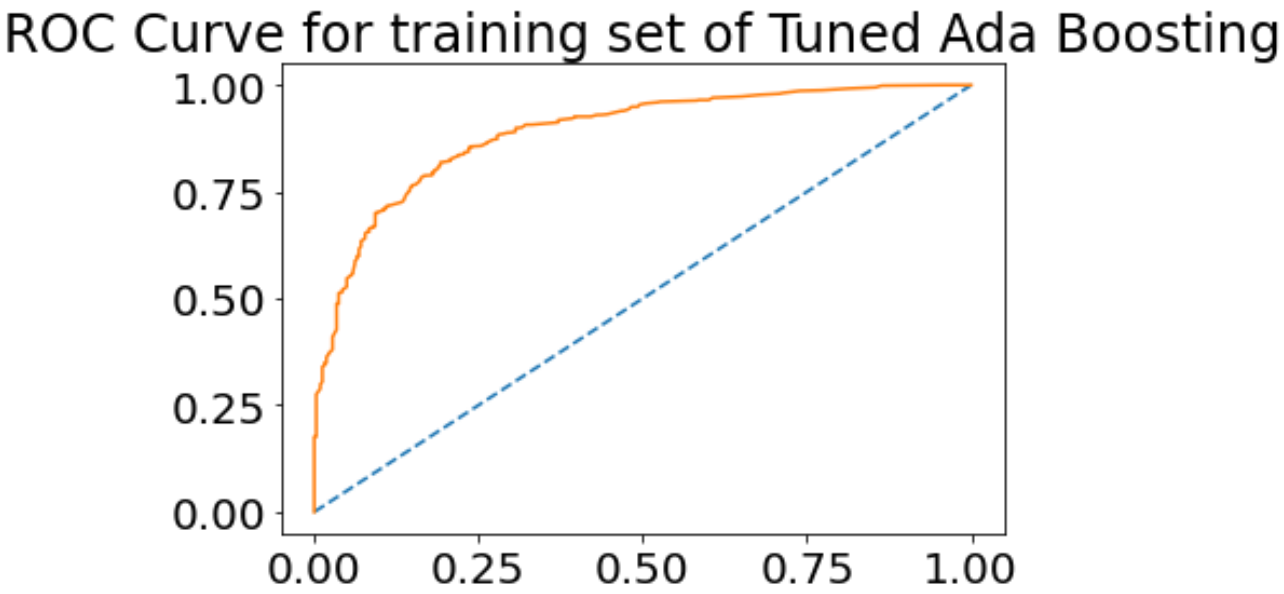

Fig. 28: ROC Curve – Tuned Ada Boost – Train

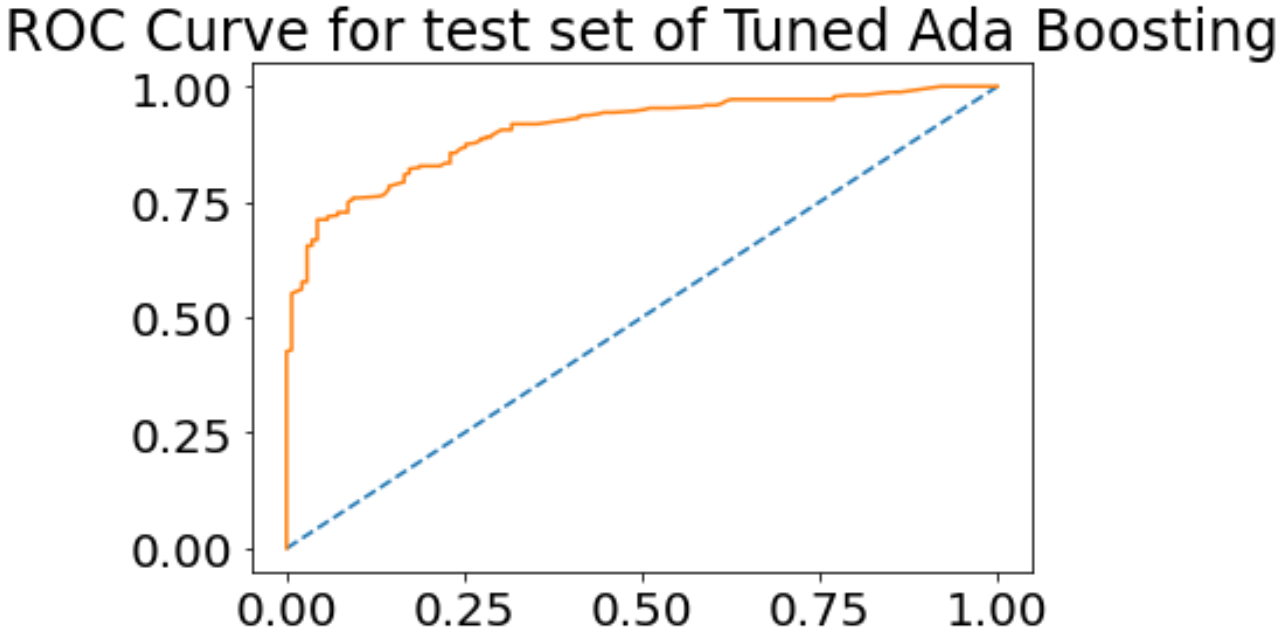

Fig. 29: ROC Curve – Tuned Ada Boost – Test

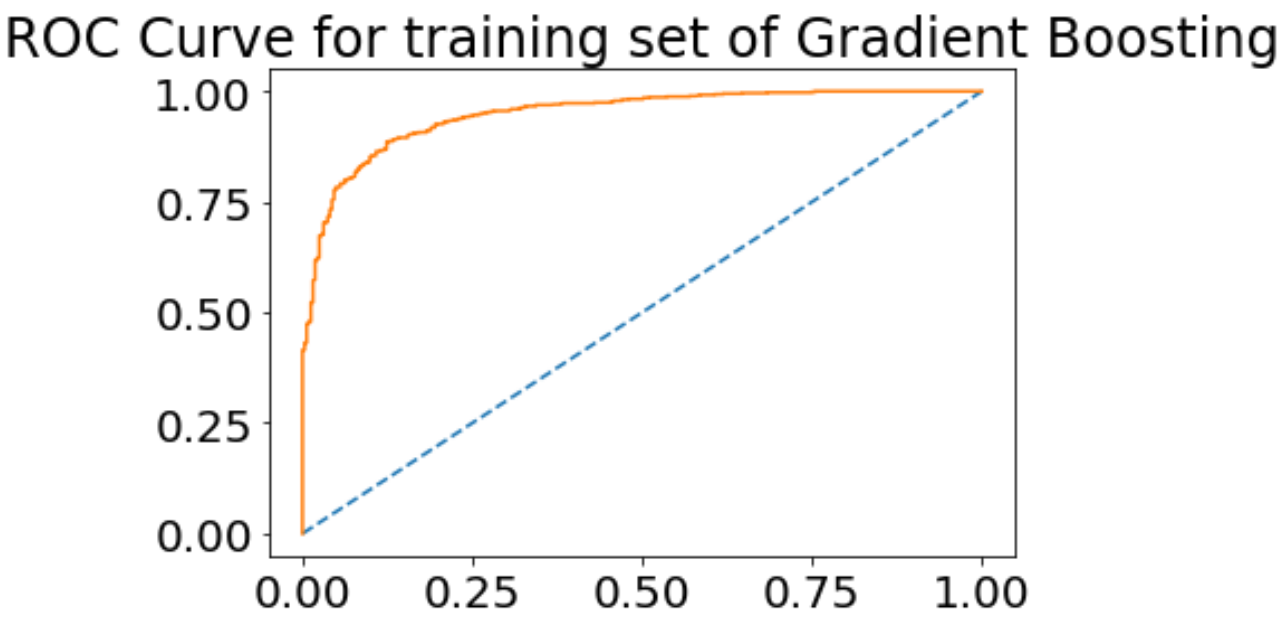

Fig. 30: ROC Curve –Gradient Boost – Train

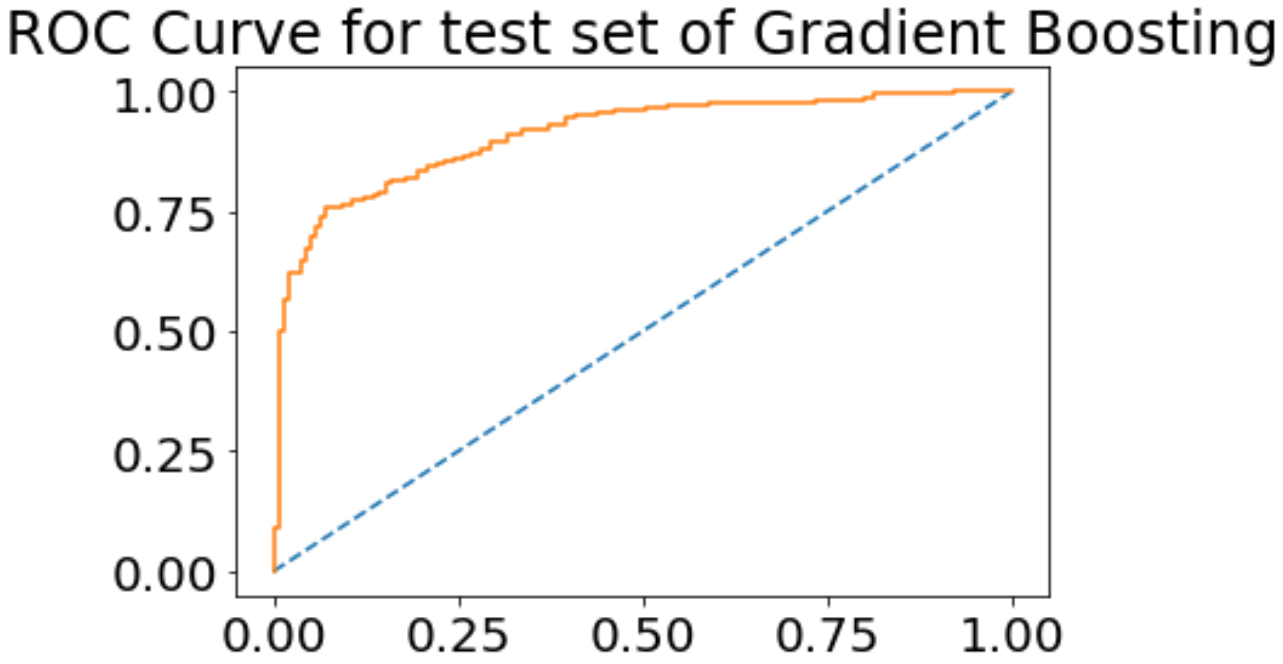

Fig. 31: ROC Curve – Gradient Boost - Test

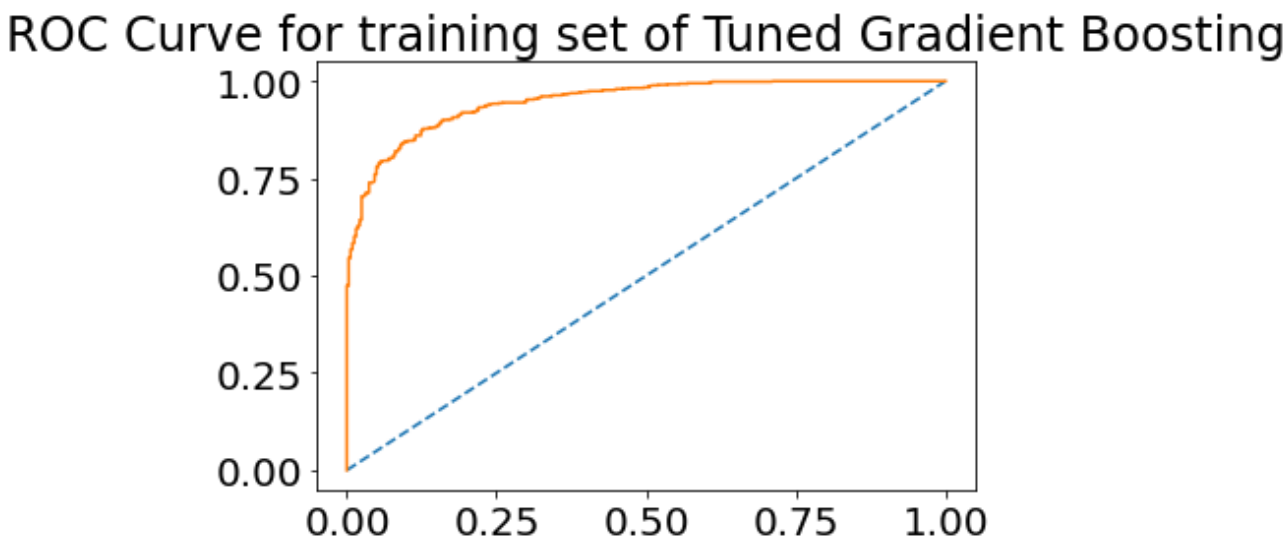

Fig. 32: ROC Curve –Tuned Gradient Boost – Train

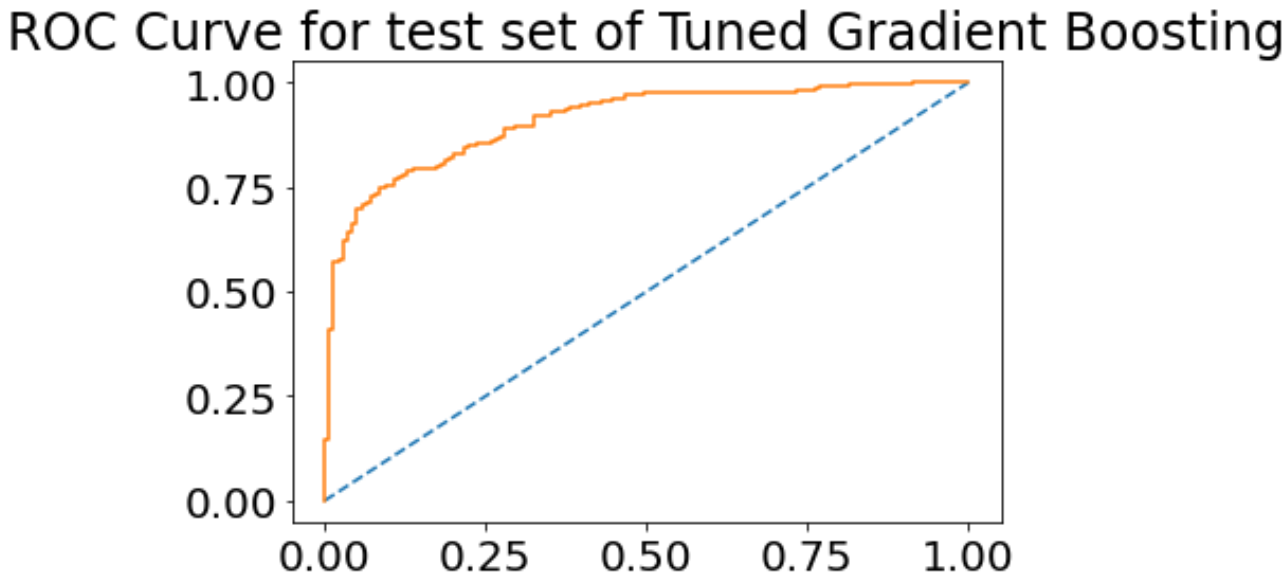

Fig. 33: ROC Curve –Tuned Gradient Boost - Train

Model	Data	Accuracy	Precision		Recall		F1 score		AUC
			Class 0	Class 1	Class 0	Clas s1	Class 0	Clas s1	
Logistic Regressio n	Train	0.83	0.75	0.86	0.65	0.91	0.70	0.88	0.877
	Test	0.85	0.80	0.87	0.68	0.92	0.73	0.90	0.914
LDA	Train	0.83	0.73	0.86	0.67	0.90	0.70	0.88	0.876
	Test	0.84	0.77	0.87	0.70	0.91	0.73	0.89	0.915
KNN	Train	0.86	0.79	0.89	0.74	0.91	0.77	0.90	0.934
	Test	0.84	0.75	0.87	0.68	0.90	0.72	0.88	0.879
Naïve Bayes	Train	0.82	0.71	0.87	0.69	0.88	0.70	0.87	0.874
	Test	0.85	0.76	0.88	0.73	0.90	0.74	0.89	0.910
Tuned Logistic Regressio n	Train	0.83	0.77	0.84	0.60	0.92	0.68	0.88	0.874
	Test	0.84	0.80	0.86	0.65	0.93	0.72	0.89	0.911
Tuned KNN	Train	0.84	0.75	0.87	0.70	0.90	0.72	0.89	0.913
	Test	0.84	0.77	0.87	0.69	0.91	0.73	0.89	0.892
Tuned Random Forest	Train	0.85	0.84	0.85	0.63	0.95	0.72	0.90	0.913
	Test	0.84	0.83	0.84	0.59	0.95	0.69	0.89	0.912

Model	Data	Accuracy	Precision		Recall		F1 score		AUC
			Class 0	Class 1	Class 0	Class 1	Class 0	Class 1	
Bagging	Train	0.84	0.82	0.85	0.60	0.94	0.70	0.89	0.906
	Test	0.85	0.84	0.85	0.63	0.95	0.72	0.90	0.915
Tuned Bagging	Train	0.84	0.83	0.84	0.60	0.94	0.70	0.89	0.906
	Test	0.84	0.84	0.84	0.60	0.95	0.70	0.89	0.913
Ada Boost	Train	0.84	0.76	0.88	0.70	0.91	0.73	0.89	0.899
	Test	0.83	0.74	0.86	0.68	0.90	0.71	0.88	0.906
Tuned Ada Boost	Train	0.84	0.75	0.87	0.69	0.90	0.72	0.88	0.889
	Test	0.84	0.76	0.87	0.70	0.91	0.73	0.89	0.906
Gradient Boost	Train	0.89	0.85	0.90	0.76	0.94	0.80	0.92	0.948
	Test	0.84	0.77	0.86	0.66	0.92	0.71	0.89	0.908
Tuned Gradient Boost	Train	0.88	0.84	0.90	0.77	0.94	0.80	0.92	0.949
	Test	0.84	0.78	0.87	0.68	0.92	0.72	0.89	0.909

Table 35: Summary of all models.

When we compare the accuracies, we see that logistic regression model is performing better on the test data. However, since we have data imbalance in the target column, we can't rely on the accuracy scores. In this case, both the target classes are important to us. Hence, precision and recall both carry same weightage in our problem. Hence, we will consider f1 scores to compare our models. We find that KNN and Gradient boost models performing better than others. When we consider the AUC scores, LDA and bagging models are performing better than other models.

1.8) Based on your analysis and working on the business problem, detail out appropriate insights and recommendations to help the management solve the business objective. There should be at least 3-4 Recommendations and insights in total. Recommendations should be easily understandable and business specific, students should not give any technical suggestions. Full marks should only be allotted if the recommendations are correct and business specific.

Insights and Recommendations:

- The data is highly imbalanced in terms of the vote. About 70% of the voters are the supporters of Labour party.
- The survey included very less people who have high political knowledge.
- The performance of the models that we have built differ when we take into account, different metrics.
- Younger people are more likely to vote for the Labour party.
- The news channel should work on collecting more data so as to balance out the points in vote column.
- Collecting more data may help in building a better performing model.

Part -2

Text Analytics

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

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

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

The number of Characters, words and sentences in each speech documents are as given below.

Name of the President	Number of characters	Number of words	Number of sentences
Roosevelt	7571	1536	68
Kennedy	7618	1546	52
Nixon	9991	2028	69

Table 36: Number of characters, words and sentences in each speech document

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.

Basic text data cleaning has been carried out. We have removed all the punctuations in the documents using the list of punctuations from the string library. All the stopwords such as ‘a’, ‘an’, ‘the’, etc have been removed from the documents using the stopwords corpus from the nltk library. We have additionally removed the symbol double hyphen (--) from the documents. We have also removed the word ‘Mr.’ from the documents for we believe it adds no value to the analysis.

After removing all the stopwords, the number of words in Roosevelt’s speech document is 624, the number of words in Kennedy’s speech document is 687 and the number of words in Nixon’s speech document is 816,

Name of the President	Word count before preprocessing	Word count after preprocessing
Roosevelt	1536	620
Kennedy	1546	685
Nixon	2028	815

Table 37: Number of words before and after removing stop words.

Sample:

'On each national day of inauguration since 1789, the people have renewed their sense of dedication to the United States.\n\nIn Washington\'s day the task of the people was to create and weld together a nation.\n\nIn Lincoln\'s day the task of the people was to preserve that Nation from disruption from within.\n\nIn this day the task of the people is to save that Nation and its institutions from disruption from without.\n\nTo us there has come a time, in the midst of swift happenings, to pause for a moment and take stock -- to recall what our place in history has been, and to rediscover what we are and what we may be. If we do not, we risk the real peril of inaction.\n\nLives of nations are determined not by the count of years, but by the lifetime of the human spirit. The life of a man is three-score years and ten: a little more, a little less. The life of a nation is the fullness of the measure of its will to live.\n\nThere are men who doubt this. There are men who believe that democracy, as a form of Government and a frame of life, is limited or measured by a kind of mystical and artificial fate that, for some unexplained reason, tyranny and slavery have become the surging wave of the future -- and that freedom is an ebbing tide.\n\nBut we Americans

Sample: Speech document before preprocessing.

'vice presid speaker chief justic senat cook mr eisenhow fellow citizen great good countri share togethernnwhen met four year ago america bleak spirit depress prospect seemingli endless war abroad destruct conflict homenna meet today stand threshold new era peac worldnnth central question us shall use peac let us resolv era enter postwar period often time retreat isol lead stagnat home invit new danger abroadnnlet us resolv becom time great respons greatli born renew spirit promis america enter third centuri nationnnthi past year saw farreach result new polici peac continu revit tradit friendship mission peke moscow abl establish base new durabl pattern relationship among nation world america bold initi long rememb year greatest progress sinc end world war ii toward last peac worldnnth peac seek world flimsi peac mere interlud war peac endur gener comennit import understand necess limit america role maintain peacennunless america work preserv peac peacennunless america work preserv freedom freedomnnbut let us clearli understand new natur america role result new polici adopt past four yearsnnw shall respect treati commitmentsnnw shall support vigor principl countri right impos rule anoth forcennw shall continu era negoti work limit nuclear arm reduc dang

Sample: Speech document after preprocessing.

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)

After removing all the stop words, the following are the top three words in each speech document:

Name of the President	Most used words.	
	Word	Frequency
Roosevelt	nation	16
	know	10
	life	8
Kennedy	Us	11
	Let	11
	power	9
Nixon	Us	26
	america	19
	respons	16

Table 38: Most used words in the speech document

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

After removing the stopwords, the word cloud of each document is given below.

Fig 34: Word cloud of presidential speech by Roosevelt.

Fig 35: Word cloud of presidential speech by Kennedy.

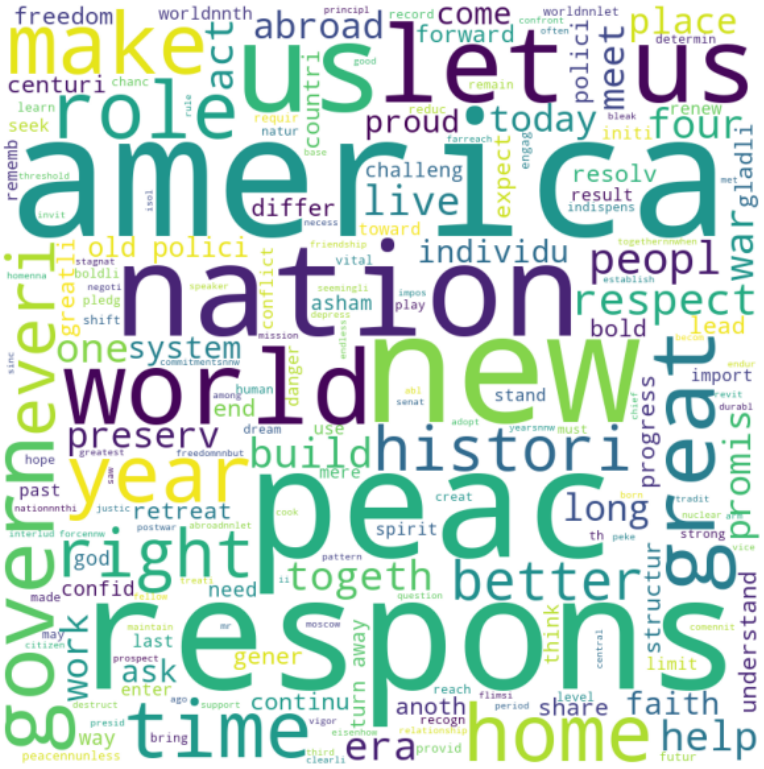

Fig 36: Word cloud of presidential speech by Nixon.