BUSINESS REPORT

Name : VIKRAM RADHAKRISHNAN

Date : 19th July ,2020

Module : Machine Learning

Problem 1:

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

Dataset for Problem: Election_Data.xlsx

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

The dataset is read from the excel sheet. The excel sheet contains two sheets within it, and the correct sheet is to be chosen.

```
df = pd.read_excel("Election_Data.xlsx",sheet_name='Election_Dataset_Two Classes')
```

It is observed that the column called "Unnamed: 0" is not necessary for the analysis and is dropped.

```
df.drop("Unnamed: 0",axis=1,inplace=True)
```

Null values are checked for and none are found

```
df.isnull().sum()
```

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

Duplicates are checked for – it is observed that some duplicates are present but these records are retained as its possible that these represent real cases.

The shape dataframe is checked

```
df.shape
(1525, 9)
```

and the type of information contained in it is queried :

	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	NaN	NaN	NaN	54.1823	15.7112	24	41	53	67	93
economic.cond.national	1525	NaN	NaN	NaN	3.2459	0.880969	1	3	3	4	5
economic.cond.household	1525	NaN	NaN	NaN	3.14033	0.929951	1	3	3	4	5
Blair	1525	NaN	NaN	NaN	3.33443	1.17482	1	2	4	4	5
Hague	1525	NaN	NaN	NaN	2.74689	1.2307	1	2	2	4	5
Europe	1525	NaN	NaN	NaN	6.72852	3.29754	1	4	6	10	11
political.knowledge	1525	NaN	NaN	NaN	1.5423	1.08331	0	0	2	2	3
gender	1525	2	female	812	NaN	NaN	NaN	NaN	NaN	NaN	NaN

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

Vote and Gender are objects as they are categorical. Similarly, most of the other features/columns also exhibit categorical behavior, except ‘Age’ which is a continuous variable.

Label encoding is performed on these features and converted to numerical values subsequently.

```
df.vote.value_counts(normalize=True)
```

```
Labour          0.697049
Conservative    0.302951
```

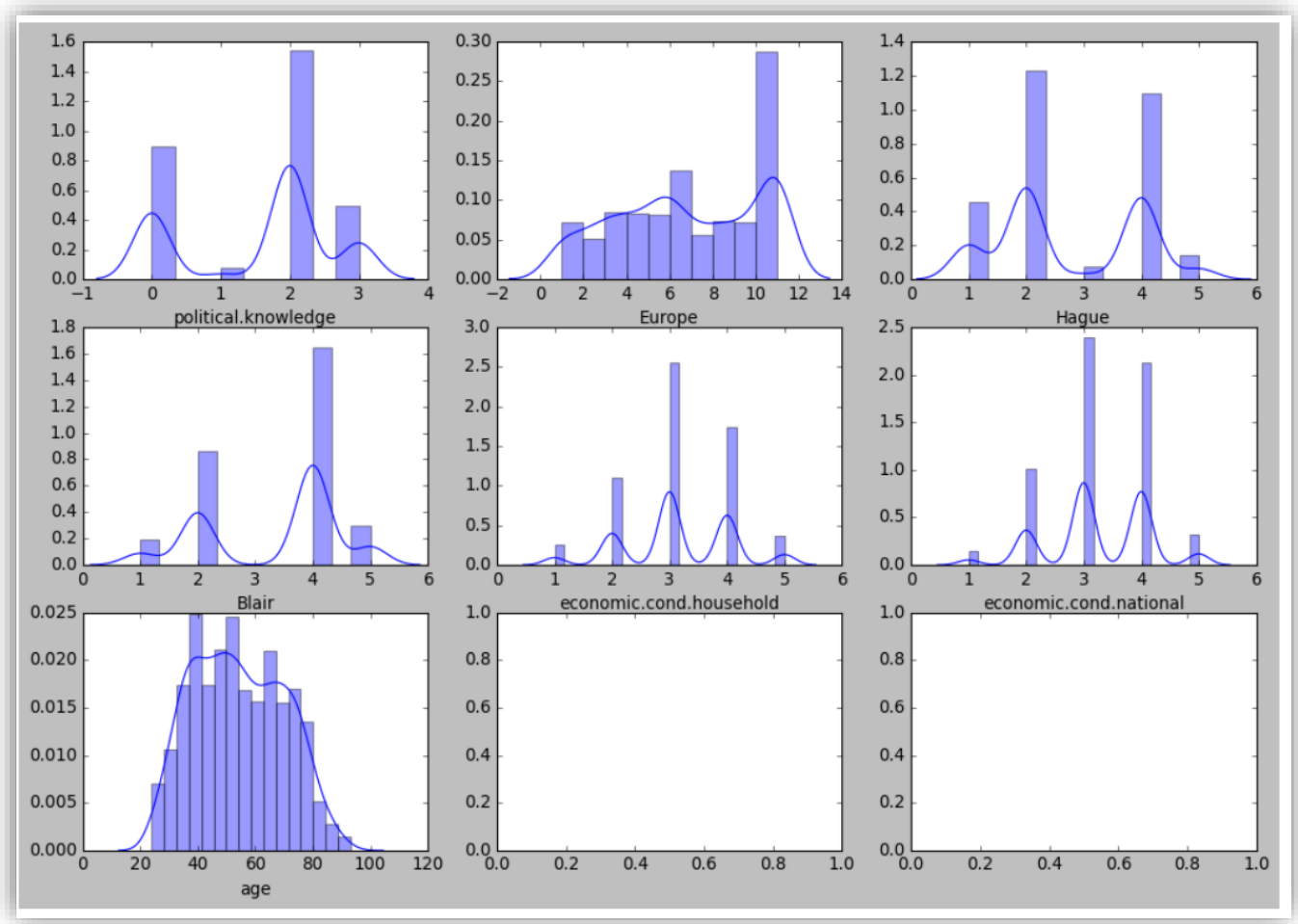
Similarly, other columns such as gender can be checked to get a general idea of the data and how is divided.

```
df.gender.value_counts(normalize=True)
```

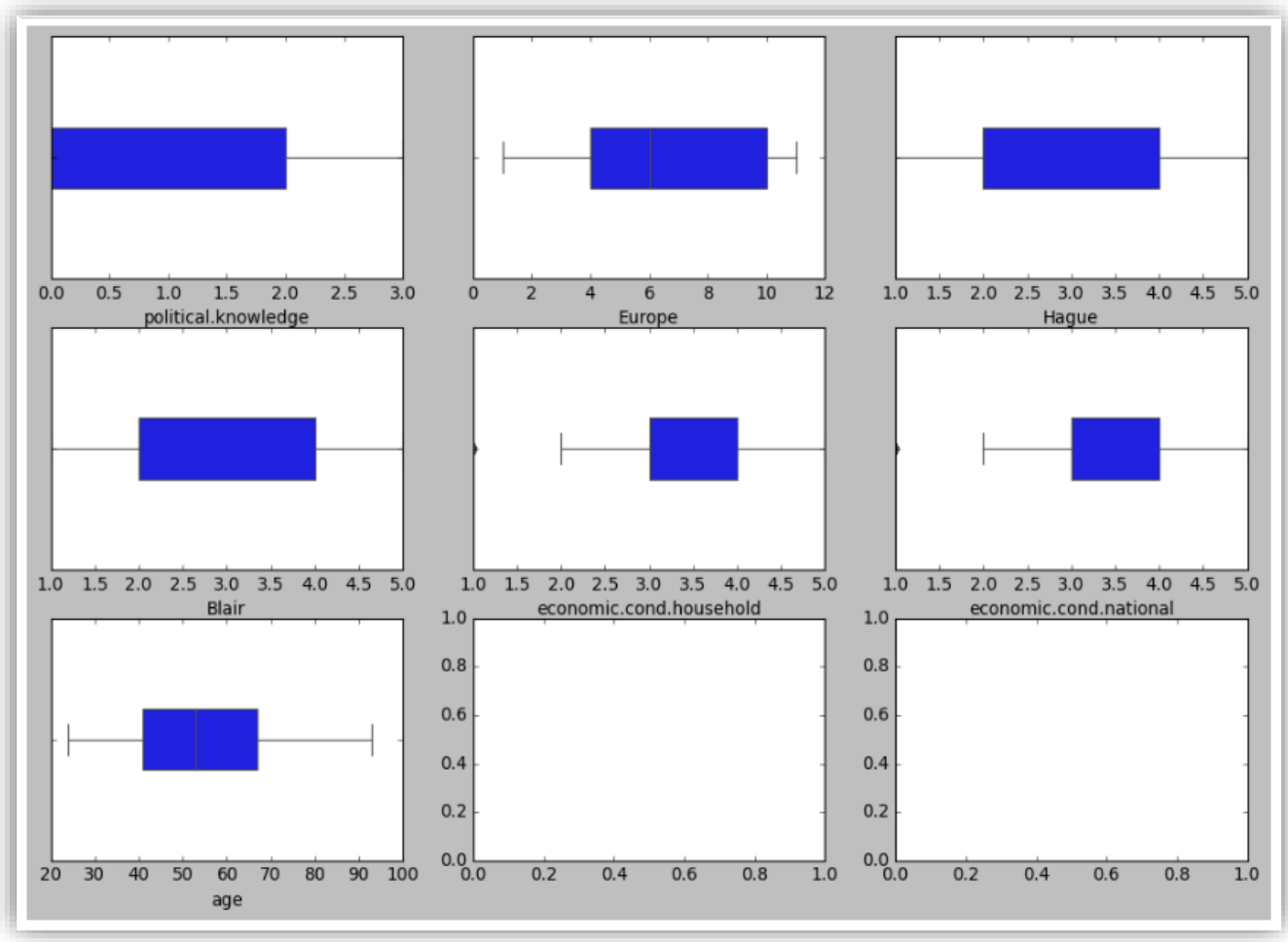
```
female    0.532459
male      0.467541
```

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

The numerical features are plotted using a distribution plot. Since most of the features are categorical (they are part of a scale/rating), many are seen as individual towers.

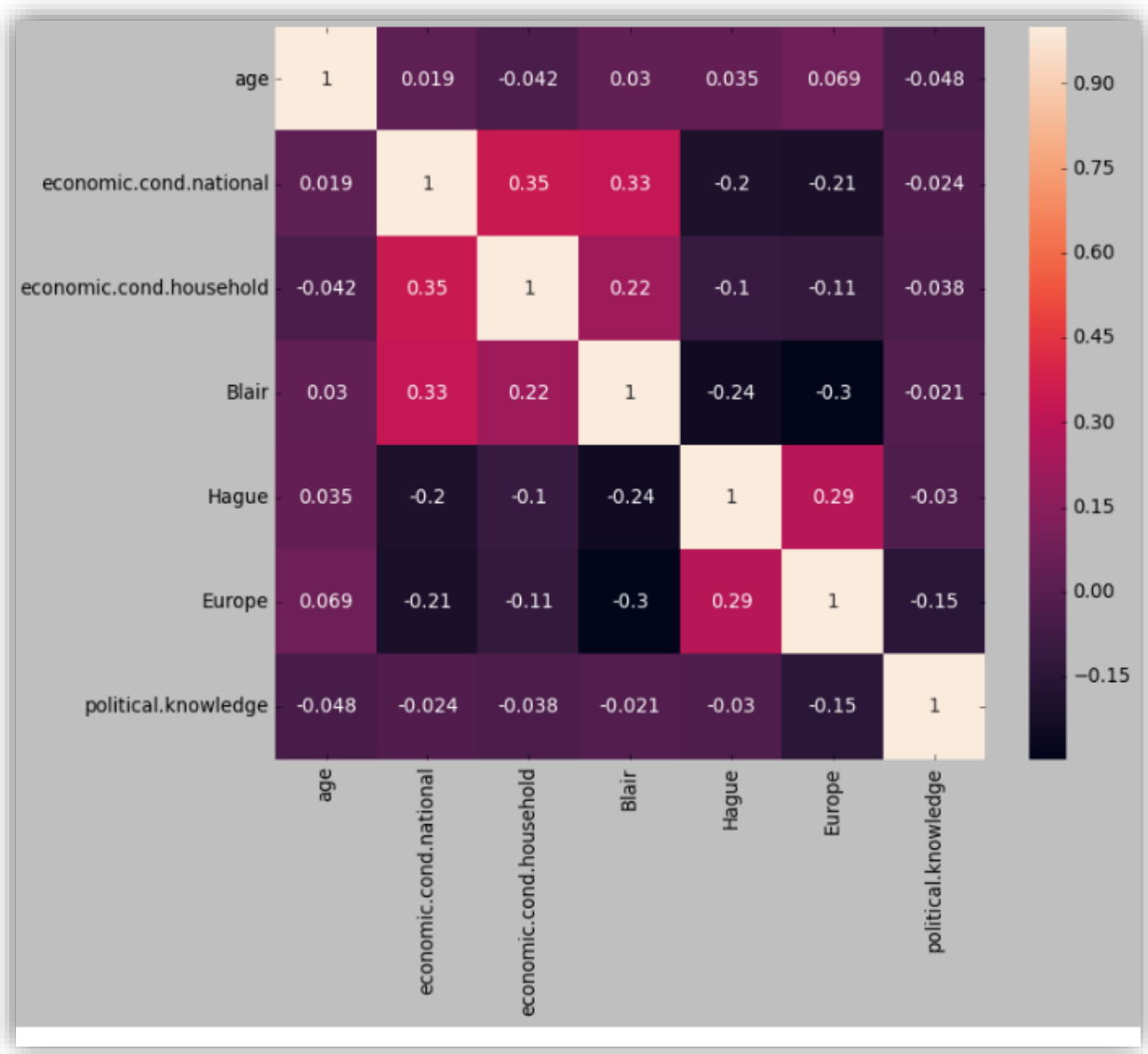


The only continuous variable, ‘age’ shows close to normal behavior as observed by the graph and therefore requires no further treatment except checking for outliers.



Outliers are checked for and its observed that the numerical/continuous columns do not have any outliers. The barplots do show outlier like dots for a few variables, but those are categorical and therefore can be ignored for outlier checks.

The correlation heatmap among variables is checked and its observed that no variables are highly correlated with each other. This also means that the features are fairly unique and none need to be dropped for the analysis at present.



Data Preparation:
1. 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).

The columns having string values which are also categorical are vote and gender. These are label encoded as shown below.

```
feature: vote
[Labour, Conservative]
Categories (2, object): [Conservative, Labour]
[1 0]


feature: gender
[female, male]
Categories (2, object): [female, male]
[0 1]
```

A quick check of the dataframe currently present is all numerical and looks like the following :

```
df.head(3)
```

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	1	43	3	3	4	1	2	2	0
1	1	36	4	4	4	4	5	2	1
2	1	35	4	4	5	2	3	2	1

The data is then split into train and test data, with 70% of the data going into training and 30% of the data being used for testing.

```
# Split X and y into training and test set in 70:30 ratio
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3 , random_state=1)
```

SCALING:

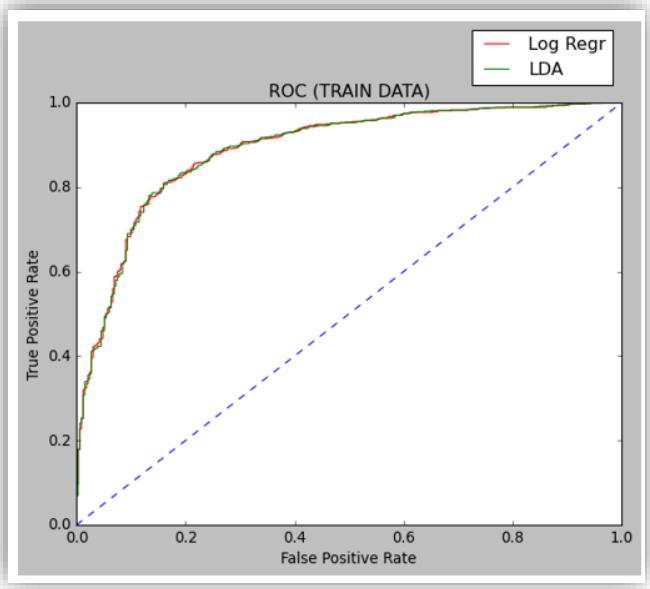
In this dataset, most of the values are categorical except age. Using encoding, all variables are converted to numerical values. Scaling is generally recommended when the features have large varying ranges of numerical values, which causes one or more features to have more weightage or skew than the others.

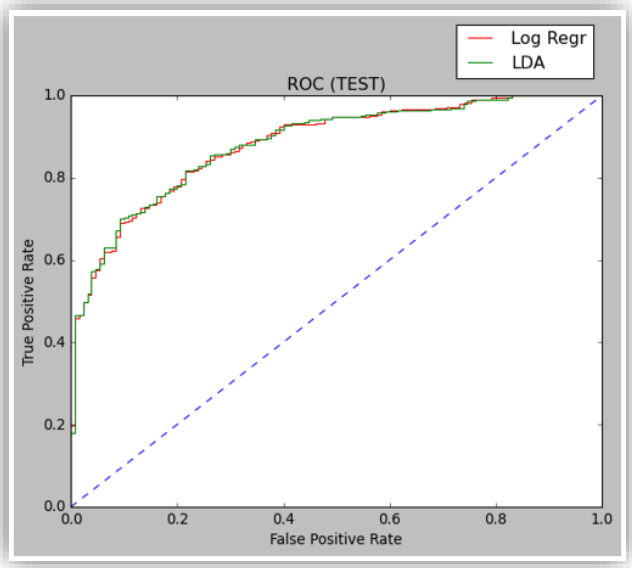
For certain classifiers such as SVM (Support Vector Machines), scaling is recommended. Here, the main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. In certain situations, scaling also speeds up the classification process in terms of computational time.

In this case, since we are dealing with multiple classifiers and algorithms, it is recommended that the data is scaled.

Modelling: 26 marks
1. Apply Logistic Regression and LDA (linear discriminant analysis).

Logistic regression and LDA are applied to the data for predictions and the behaviors are found to be similar for both test and train data



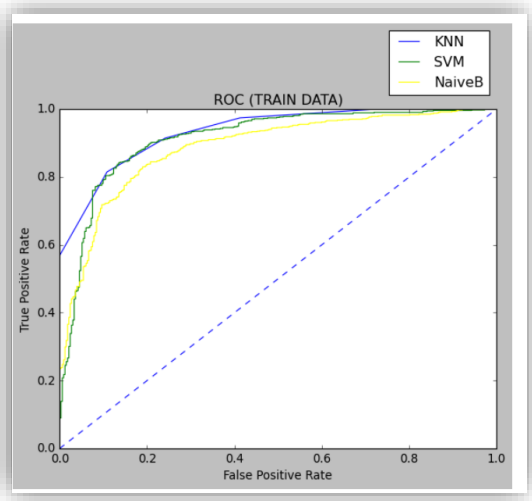


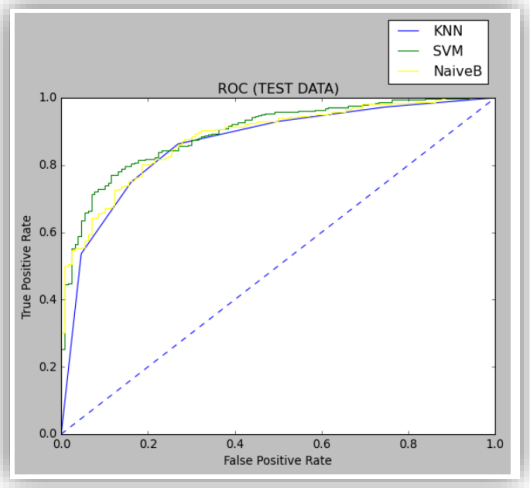
	Logit Train	Logit Reg Test	LDA Train	LDA Test
Accuracy	0.84	0.82	0.84	0.82
AUC	0.89	0.88	0.89	0.88
Recall	0.91	0.89	0.90	0.88
Precision	0.87	0.87	0.87	0.87
F1 Score	0.89	0.88	0.88	0.87

These two models also perform reasonably well on other metrics such as accuracy, recall, precision and score and therefore work well with this data.

2. Apply KNN Model, Naïve Bayes Model and support vector machine (SVM) model. Interpret the results. (7 marks)

The ROC curves for the 3 classifiers or algorithms are drawn together for test and train data all of them show similar behavior for test and train.





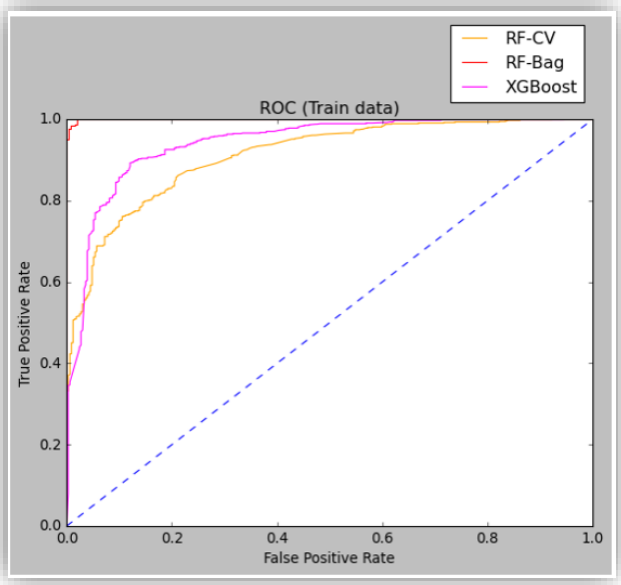
	KNN Train	KNN Test	SVM Train	SVM Test	NB Train	NB Test
Accuracy	0.87	0.83	0.86	0.82	0.83	0.83
AUC	0.94	0.86	0.91	0.90	0.89	0.88
Recall	0.91	0.86	0.93	0.90	0.88	0.87
Precision	0.89	0.89	0.88	0.86	0.88	0.89
F1 Score	0.90	0.88	0.90	0.88	0.88	0.88

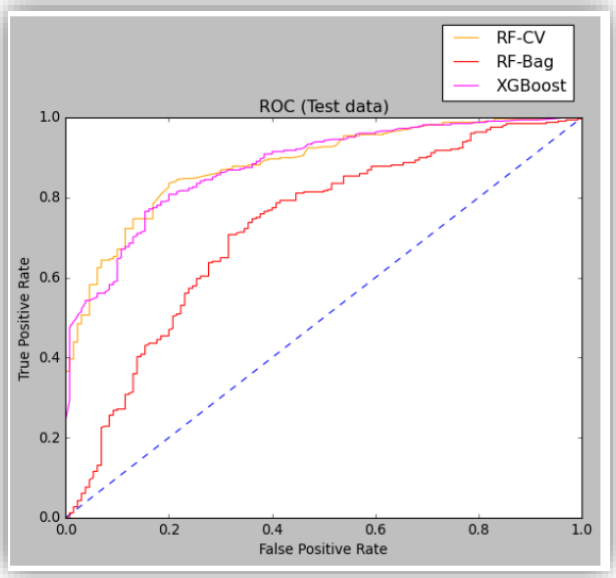
In terms of the parameters such as accuracy, recall, precision and f1 score, the there models compared above show almost similar behavior with more than 80% accuracy and therefore are all good models for this data.

These models can be further tuned to improve the accuracy and other metrics if required. However, for this case study, we are stopping here.

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

The following ROC curves are observed for training and test data using the 3 methods mentioned above.





	Tuning(RF) Train	Tuning(RF) Test	Bagging(RF) Train	Bagging(RF) Test	XG-Boost Train	XG-Boost Test
Accuracy	0.84	0.82	0.99	0.72	0.89	0.81
AUC	0.91	0.88	1.00	0.72	0.94	0.88
Recall	0.91	0.88	0.98	0.99	0.93	0.85
Precision	0.87	0.87	0.99	0.72	0.91	0.88
F1 Score	0.89	0.87	0.99	0.84	0.92	0.87

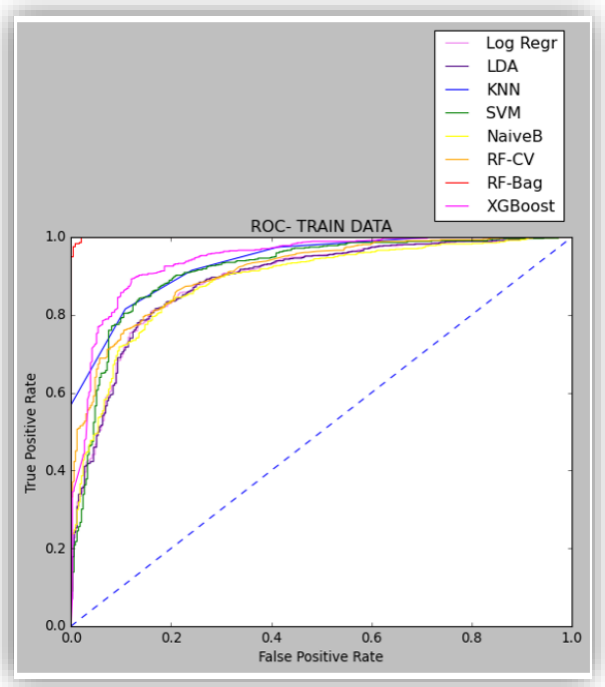
For demonstrating tuning, we use the Random Forest Classifier and vary both the cross validation (CV) as well as certain hyperparameters through a ‘Grid-Search’ in the attempt to obtain better accuracy.

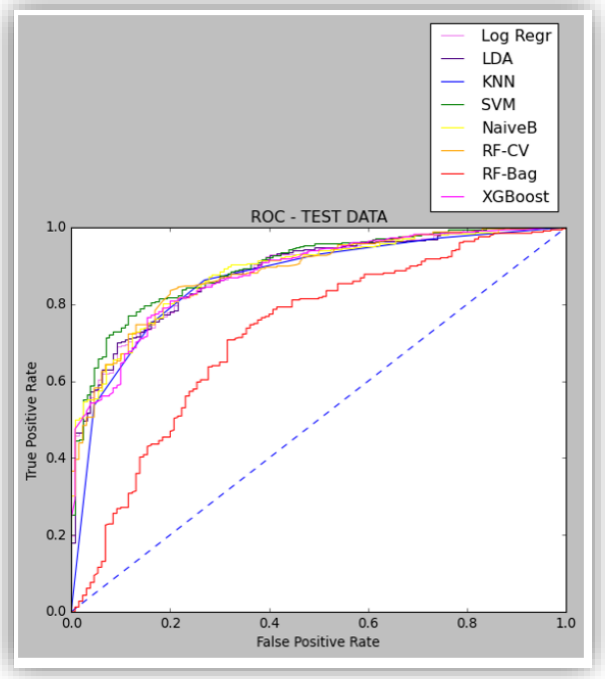
In the second case, with the random forest classifier, we also attempt Bagging (Ensemble methods). In this case, we observe “overfitting” in the training data and poor performance in the test data. Therefore, we have two options – to improve the performance of the current method by tuning the classifier or discarding it as other classifiers show promise for this data. For now, we do not further tune this model.

In the third case, we are to use boosting and we have a choice between AdaBoost and XGBoost. Here, we go for XGBoost because of its robust nature and quick processing time. Its found that the XGBoost algorithm performs well, on par with logit, LDA, SVM, KNN and NB methods.

Therefore, the random forest method with tuning and the XGBoost method are OK to use with this particular data. The bagging method needs to be tuned further to prevent overfitting and this can be done in many ways such as feature reduction, using tuning to get a more agreeable test accuracy that’s inline with test predictions.

4. 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. (7 marks)





	Logit Train	Logit Reg Test	LDA Train	LDA Test	KNN Train	KNN Test	SVM Train	SVM Test	NB Train	NB Test	Tuning(RF) Train	Tuning(RF) Test	Bagging(RF) Train	Bagging(RF) Test	XGBoost Train	XGBoost Test
Accuracy	0.84	0.82	0.84	0.82	0.87	0.83	0.86	0.82	0.83	0.83	0.84	0.82	0.99	0.72	0.89	0.81
AUC	0.89	0.88	0.89	0.88	0.94	0.86	0.91	0.90	0.89	0.88	0.91	0.88	1.00	0.72	0.94	0.88
Recall	0.91	0.89	0.90	0.88	0.91	0.86	0.93	0.90	0.88	0.87	0.91	0.88	0.98	0.99	0.93	0.85
Precision	0.87	0.87	0.87	0.87	0.89	0.89	0.88	0.86	0.88	0.89	0.87	0.87	0.99	0.72	0.91	0.88
F1 Score	0.89	0.88	0.88	0.87	0.90	0.88	0.90	0.88	0.88	0.88	0.89	0.87	0.99	0.84	0.92	0.87

From the table above, we can see that most of the models (with the exception of Bagging-RF) perform well and display similar metrics for parameters such as Accuracy, Recall e.t.c.

However, if we had to choose only one single model out of them, we could go for SVM on account of very minute/marginal difference in its performance and characteristics in the table.

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

```
              precision    recall  f1-score   support

         0.0       0.82      0.71      0.76       332
         1.0       0.88      0.93      0.90       735

    accuracy                           0.86      1067
   macro avg       0.85      0.82      0.83      1067
weighted avg       0.86      0.86      0.86      1067
```

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

```
              precision    recall  f1-score   support

         0.0       0.71      0.64      0.67       130
         1.0       0.86      0.90      0.88       328

    accuracy                           0.82       458
   macro avg       0.79      0.77      0.78       458
weighted avg       0.82      0.82      0.82       458
```

```
confusion_matrix(y_train, ytrain_predict)
```

```
array([[262,  70],
       [ 51, 684]], dtype=int64)
```

```
cnf_matrix=confusion_matrix(y_test, ytest_predict)
cnf_matrix
```

```
array([[ 83,  47],
       [ 34, 294]], dtype=int64)
```

Using the confusion matrices, once can further check based on TP, TN,FP, FN and the derived quantities as to which classifier is most suitable for the particular problem.

Inference:

Based on these predictions, what are the insights?

Most of the insights on the models and their behavior have already been covered. Most of the models here perform well with similar behavior, except the Bagging method which led to overfitting. However, we see marginally better behavior using SVM method, though not by a large value to call it a leader amongst the others.

The models can be improved by techniques such as GRID SEARCH and Cross Validation to further reduce the difference between train and test accuracy, and improve on other parameters such as recall, precision e.t.c

With larger number of features and changes in datatypes, we may see a difference in the performance of these models. The observation is unique only to this dataset.

The fact that machine learning methods enable one to guess the outcome of polls with an accuracy of more than 80% is quite useful. This will enable people participating in the elections to understand which areas can be improved (corresponding to features in the data) to improve the outcome of the polls.

Problem 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:

President Franklin D. Roosevelt in 1941,

President John F. Kennedy in 1961,

President Richard Nixon in 1963

- **Find the number of characters, words and sentences for the mentioned documents**

The following pictures below contain the information that has been requested (count of words, characters and sentences)

```
ROOSEVELT


The total number of characters in the speech EXCLUDING spaces and punctuations is :  5925
The total number of characters in the speech INCLUDING spaces and punctuations is :  10822
The total number of words in the speech is :  1315
The total number of sentences in the speech is :  68
```

```
NIXON


The total number of characters in the speech EXCLUDING spaces and punctuations is :  7834
The total number of characters in the speech INCLUDING spaces and punctuations is :  14210
The total number of words in the speech is :  1759
The total number of sentences in the speech is :  68
```

```
KENNEDY


The total number of characters in the speech EXCLUDING spaces and punctuations is :  5961
The total number of characters in the speech INCLUDING spaces and punctuations is :  10887
The total number of words in the speech is :  1337
The total number of sentences in the speech is :  52
```

- **Remove all the stopwords from all the three speeches.**

This has been done using the corpus from NLTK to get Stopwords in English. The following are a few command snippets used. Please refer to the code for more details.

```
stop = stopwords.words('english')
```

```
sentence_without_sw = [word for word in tokenized_text if not word in stopwords.words()]
```

```
words_separate_without_sw = [word for word in words_separate if not word in stop]
```

.

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

TOP-3 WORDS For Each President (After Removing Stop-Words) are given in the screenshots below :

ROOSEVELT

```
The three most frequent words and their occurences are :
  [('nation', 12), ('know', 10), ('spirit', 9)]
```

NIXON

```
The three most frequent words and their occurences are :
  [('let', 16), ('us', 12), ('world', 8)]
```

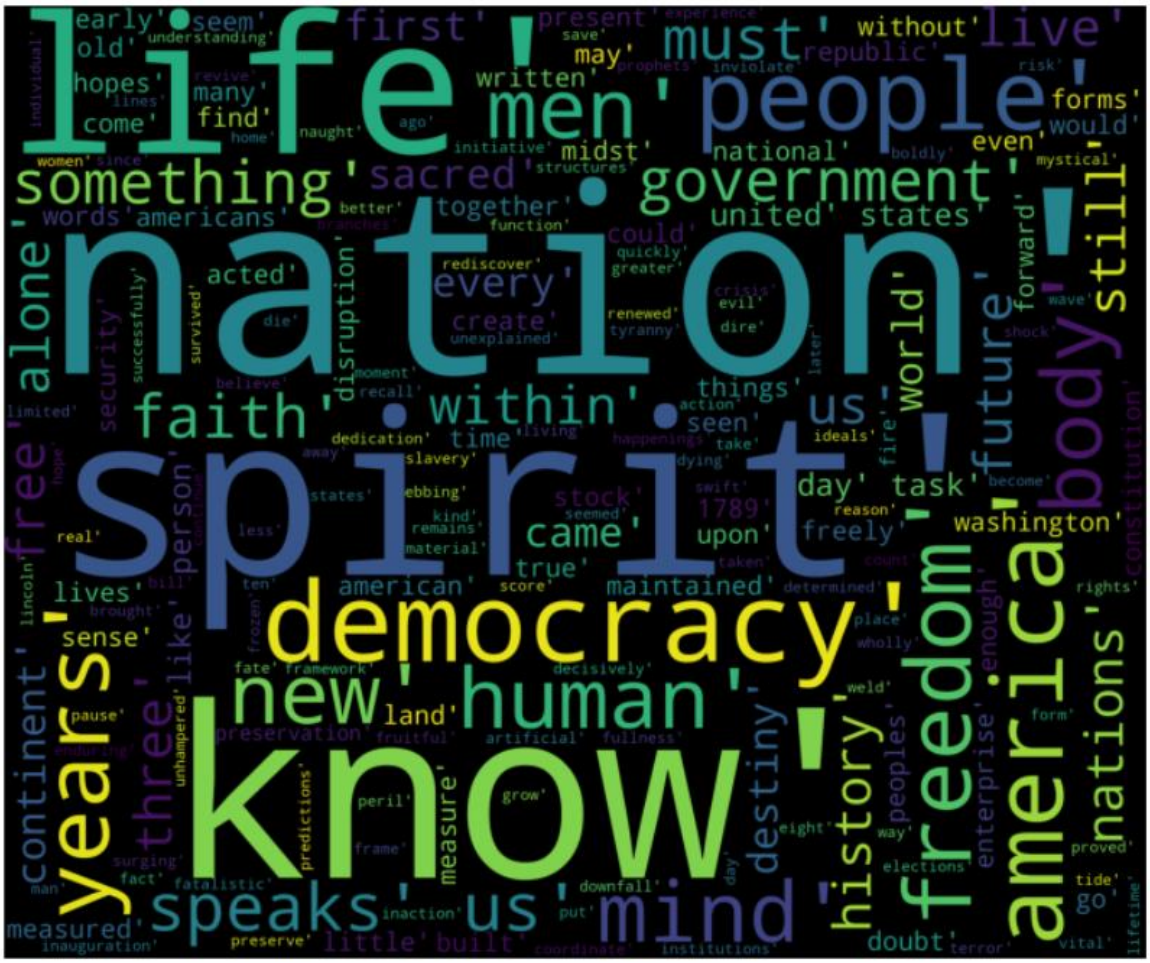
KENNEDY

```
The three most frequent words and their occurences are :
  [('us', 26), ('let', 22), ('america', 21)]
```

- Plot the word cloud of each of the speeches of the variable. (after removing the stopwords) – 3 Marks [refer to the End-to-End Case Study done in the Mentored Learning Session]

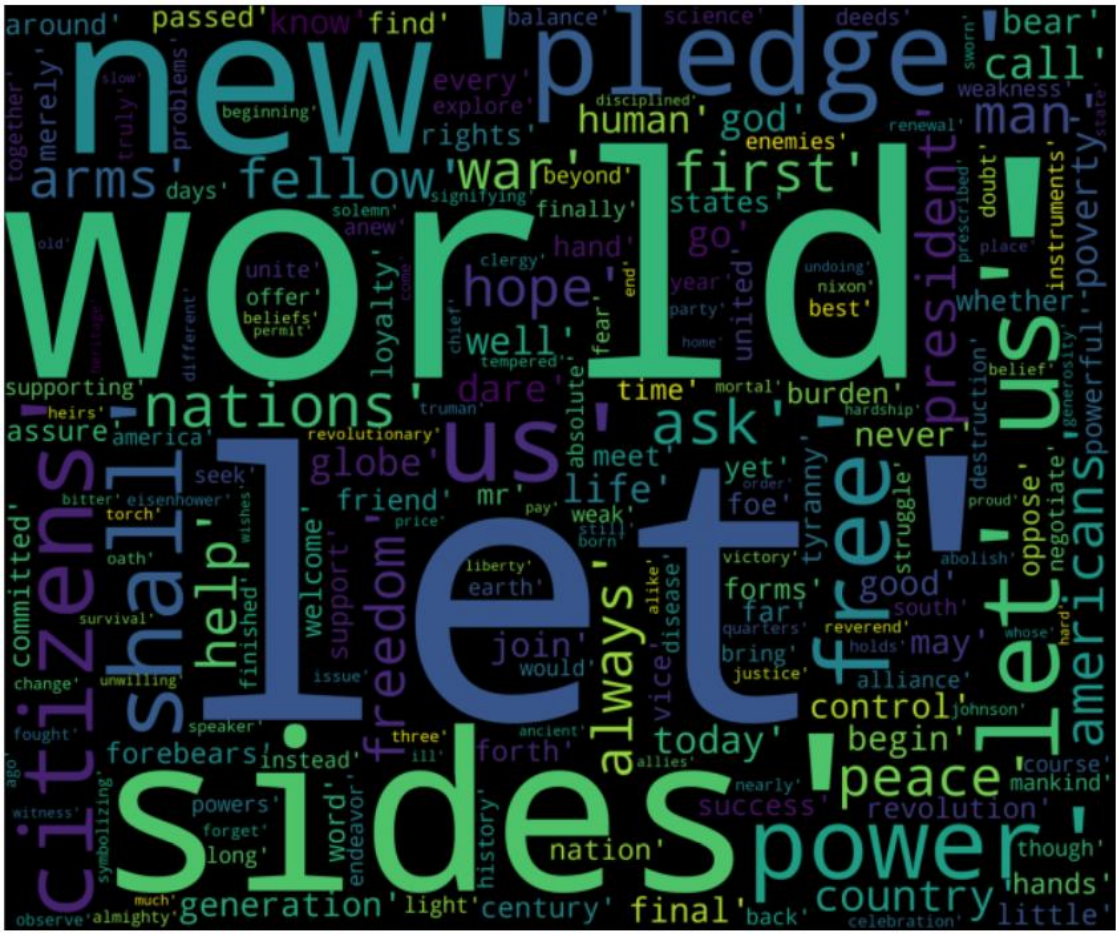
ROOSEVELT

```
The three most frequent words and their occurences are :
  [('nation', 12), ('know', 10), ('spirit', 9)]
```



NIXON

```
The three most frequent words and their occurences are :
  [('let', 16), ('us', 12), ('world', 8)]
```



KENNEDY

```
The three most frequent words and their occurences are :
  [('us', 26), ('let', 22), ('america', 21)]
```