greatlearning
Power Ahead

MACHINE LEARNING CLASSIFICATION MODEL

Mohamed Rifaz Ali K S

PGP-DSBA Online

January’ 22

Contents

List of Tables

List of Figures

Problem 1: UK Exit Poll Project

Executive Summary

A news channel CNBE who wants to analyze recent UK elections. . This survey was conducted on 1525 voters with 9 variables. As a Data Scientist, we are being asked to analyze the exit poll results to predict which party a voter will vote for on the basis of the given information. This will help in predicting overall win and seats covered by a particular party.

Introduction

The purpose of this problem is to predict the overall win and seats covered by a particular party. Two parties are Labour where Tony Blair is contesting and Conservative party where William Hague is contesting. Also, provide them with the top attributes that are most important predicting the vote. Using the machine learning classification models, we will predict the overall win and seats covered by Labour and Conservative party. We will be using the Machine learning models such as Logistic Regression, Linear Discriminant Analysis(LDA), KNN, Naïve Bayes', Bagging, Random Forest, Ada Boosting and Gradient Boosting.

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.

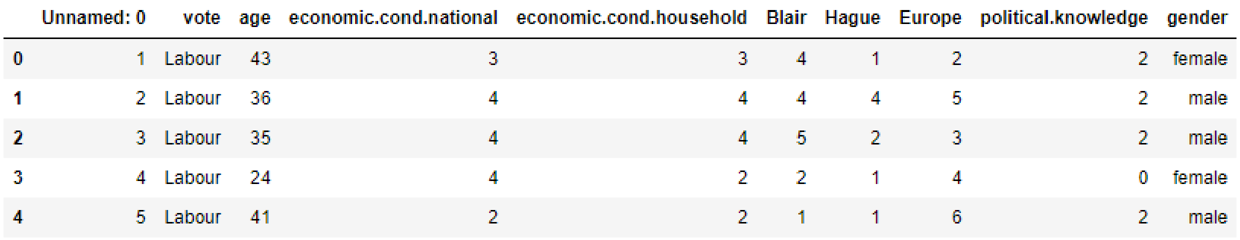
There are 1525 records and 9 columns (*excluding index column*) present in the Election Data set.

Data Description

Variable Name	**Description**
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 'Eurosceptic' sentiment.
political.knowledge	Knowledge of parties' positions on European integration, 0 to 3.
gender	female or male.

Table 1.Election Data Description

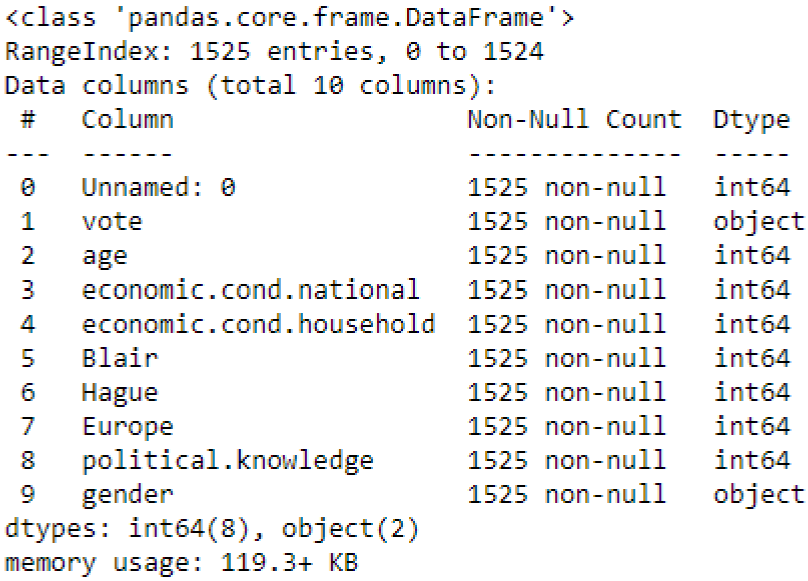
Sample of the Election Data dataset



	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 2. Sample Data Set of Election Data

Let us check the types of variables in the Data Frame



```
<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
```

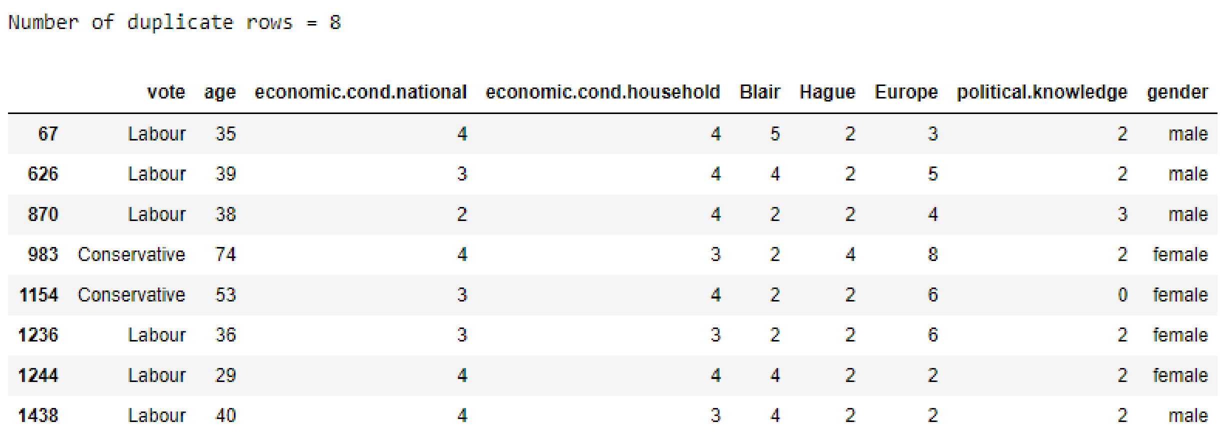
Table 3. Data Types of Election Data

We can drop the column ‘Unnamed: 0’ as it’s just an indexing to the records.

Null values check

There are no missing values present in the dataset.

Duplicate records check



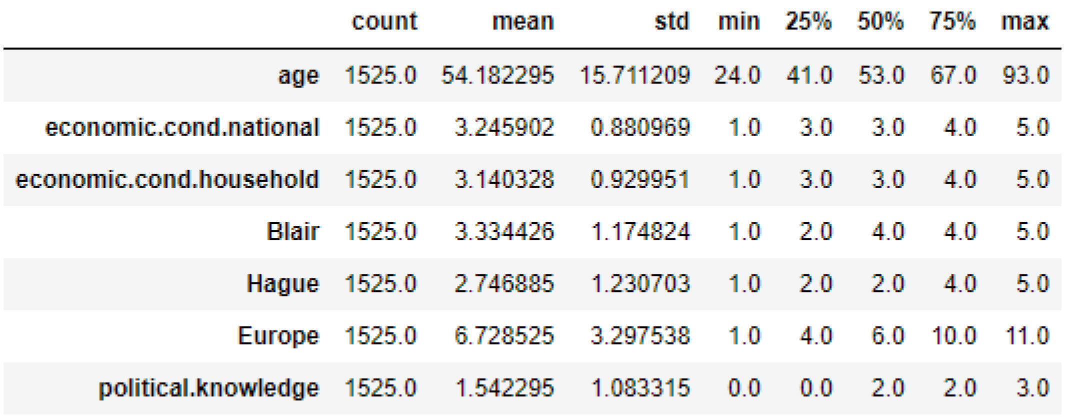
```
Number of duplicate rows = 8
```

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

Table 4. Duplicate records of Election Data

There are 8 duplicate records present in the dataset. These duplicate records may be of no use for our model building, lets drop them from the dataframe.

Summary of the Dataset



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

Table 5. Election Data Summary

- Except age column, rest of the columns is falling in the same scale.
- Almost all of the columns minimum mean and maximum values are within the quartile range, so there may be no outliers or very minimal outliers present in the dataset which is good for our model building.

Categorical columns Inspection

Column Name	Values
Vote	Labour, Conservative
Gender	Female, Male

Table 6. Categorical Columns of Election Data

Skewness and Observations

Skewness for age is: 0.14

Skewness for economic.cond.national is: -0.24

Skewness for economic.cond.household is: -0.14

Skewness for Blair is: -0.54

Skewness for Hague is: 0.15

Skewness for Europe is: -0.14

Skewness for political.knowledge is: -0.42

- The numerical columns are in ordinal in nature ranging from 1 to 5. Considering this fact, Blair is negatively skewed (*i.e., people voted high in range*) and their data's are clustered around the right side of the distribution while the left tail of the distribution is longer. On the other hand, Hague is positively skewed (*i.e., people voted low in range comparatively to Blair*).
- All the other columns are slightly negatively skewed, so the ratings are just above the average for economic conditions of the nations, houselhold and political knowledge on European integrations.

Overall summary of the Election Dataset

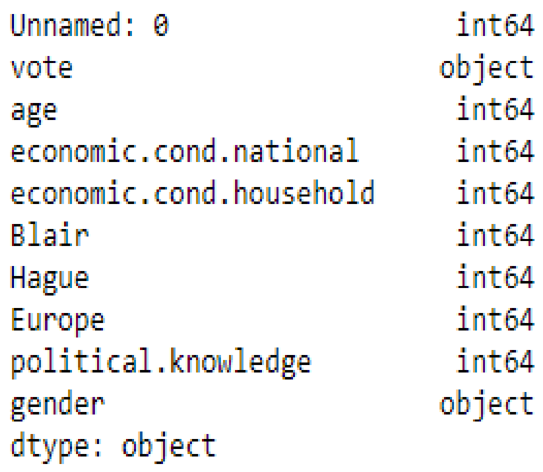
- There are 1525 records and 9 columns present in the dataset.
- Dropped 8 duplicate records from the dataset.
- There are no anomalies present in the both continuous and categorical columns.
- No null values present in the dataset.
- Age column scaling is different from other variables. We may control this through grouping the age categories.
- After dropping the duplicates, now we have 1517 records for our analysis.

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, Correlation 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.

Null values check

There are no missing values present in the dataset.

Data types

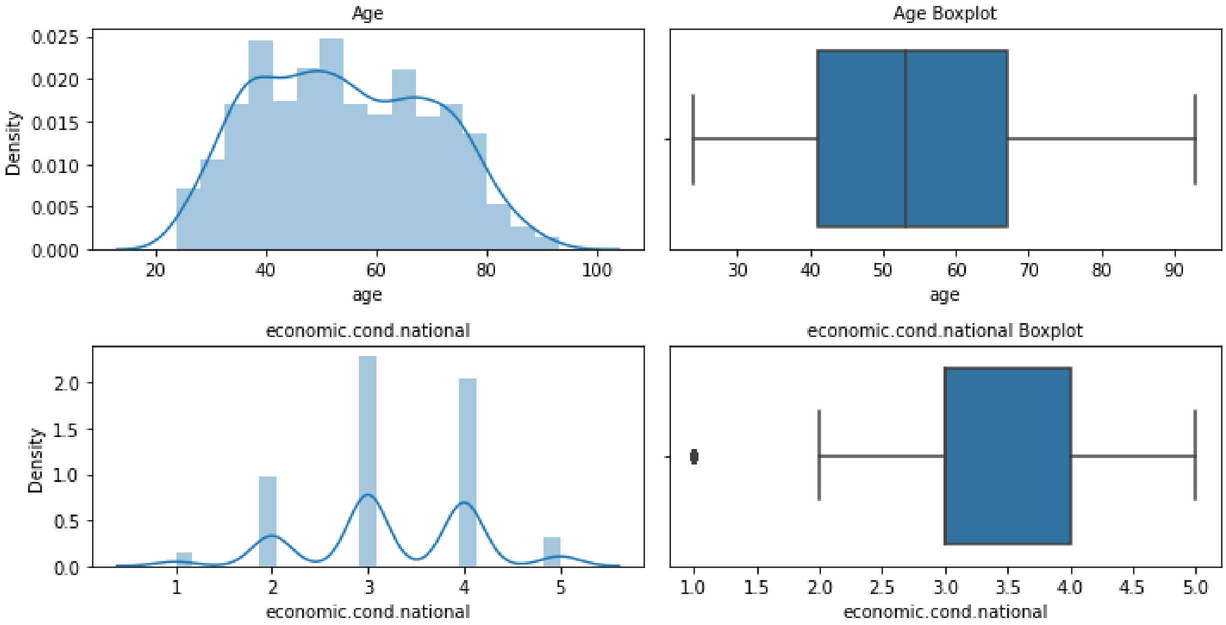


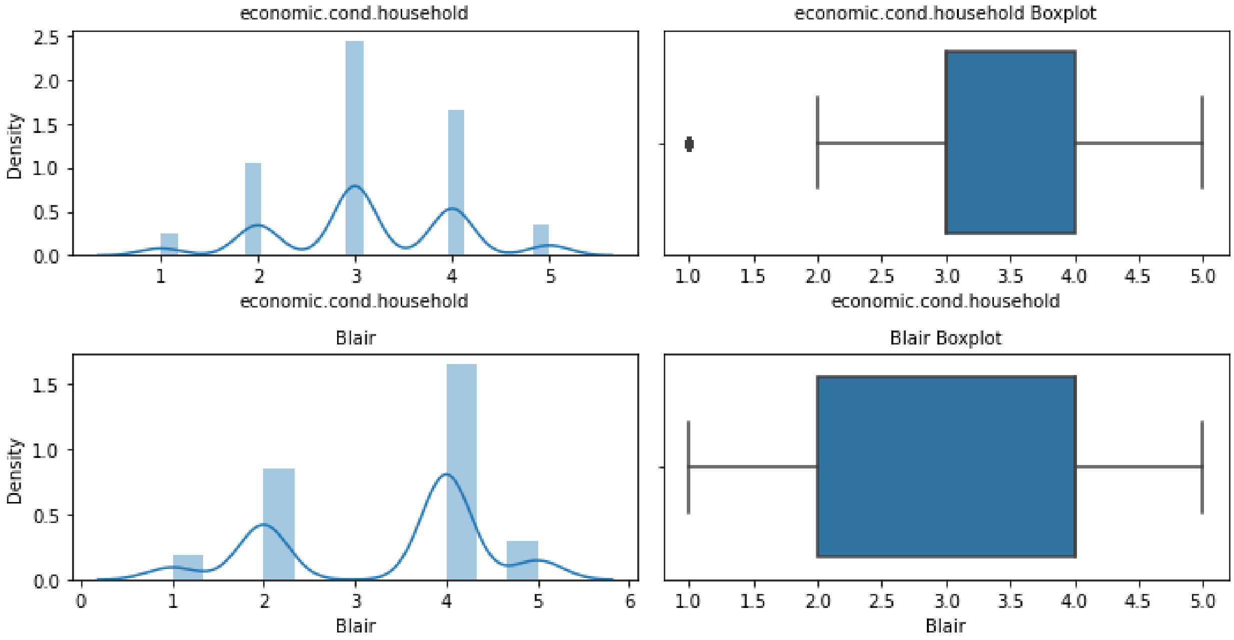
```
Unnamed: 0                  int64
vote                       object
age                         int64
economic.cond.national      int64
economic.cond.household     int64
Blair                       int64
Hague                       int64
Europe                      int64
political.knowledge     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

Table 7. Data Types of Election Data

After dropping the 8 duplicate records, there are 1517 records present in the data frame.

Univariate Analysis





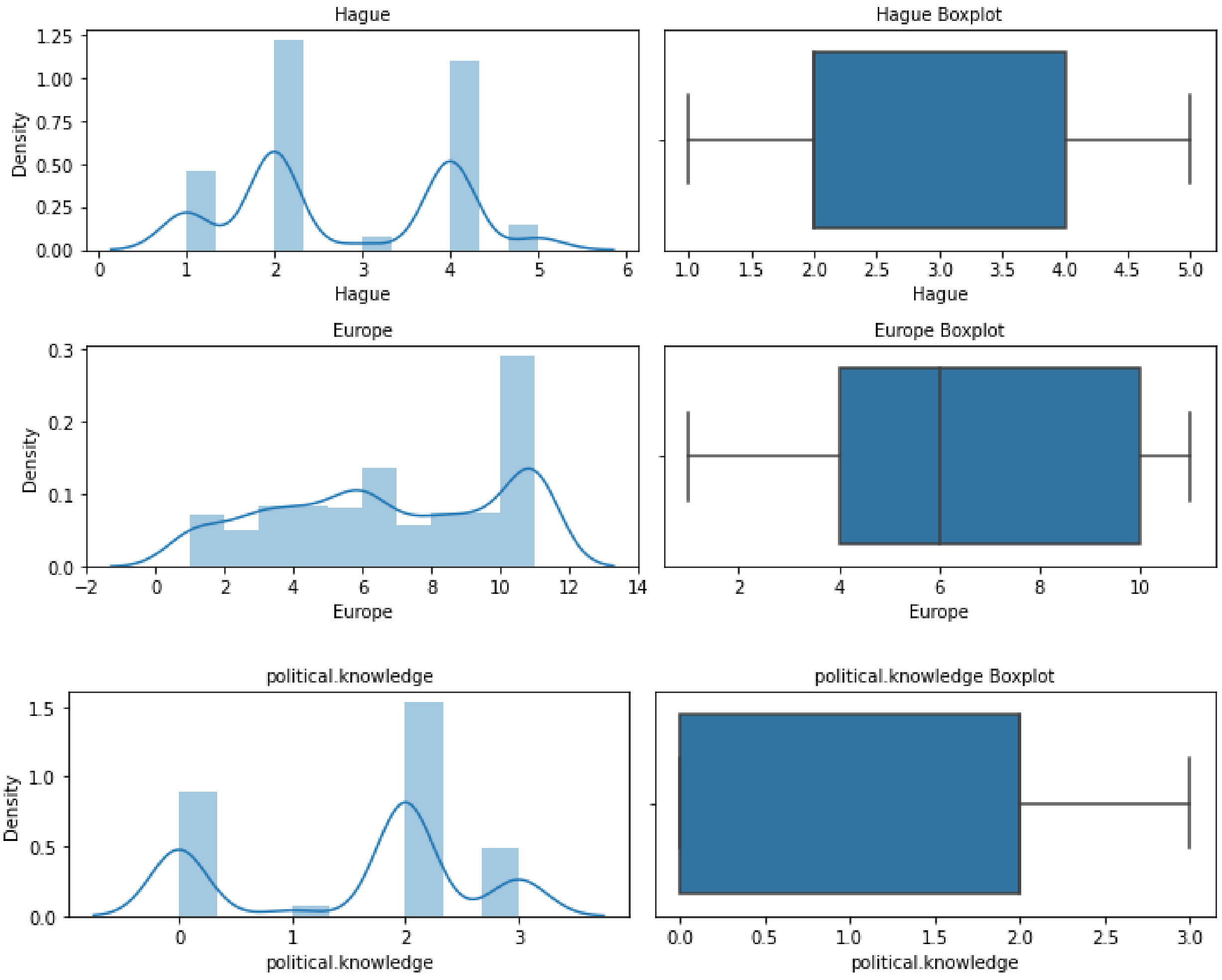


Figure 1. Histogram & Boxplot for all predictors

Observations

- Age column is normally distributed.
- As the dataset is about the exit poll survey conducted with the people in UK, all the other numeric columns are the assessment of social economic status of the nations, households and their ranges are in ordinal in nature.
- There are very few citizens who rated the current national/household economic condition is very low (i.e., 1.0), which is clearly visible in the boxplot.
- In general, people rated in the averages for all the other columns.

Pairplot and Correlation Plot

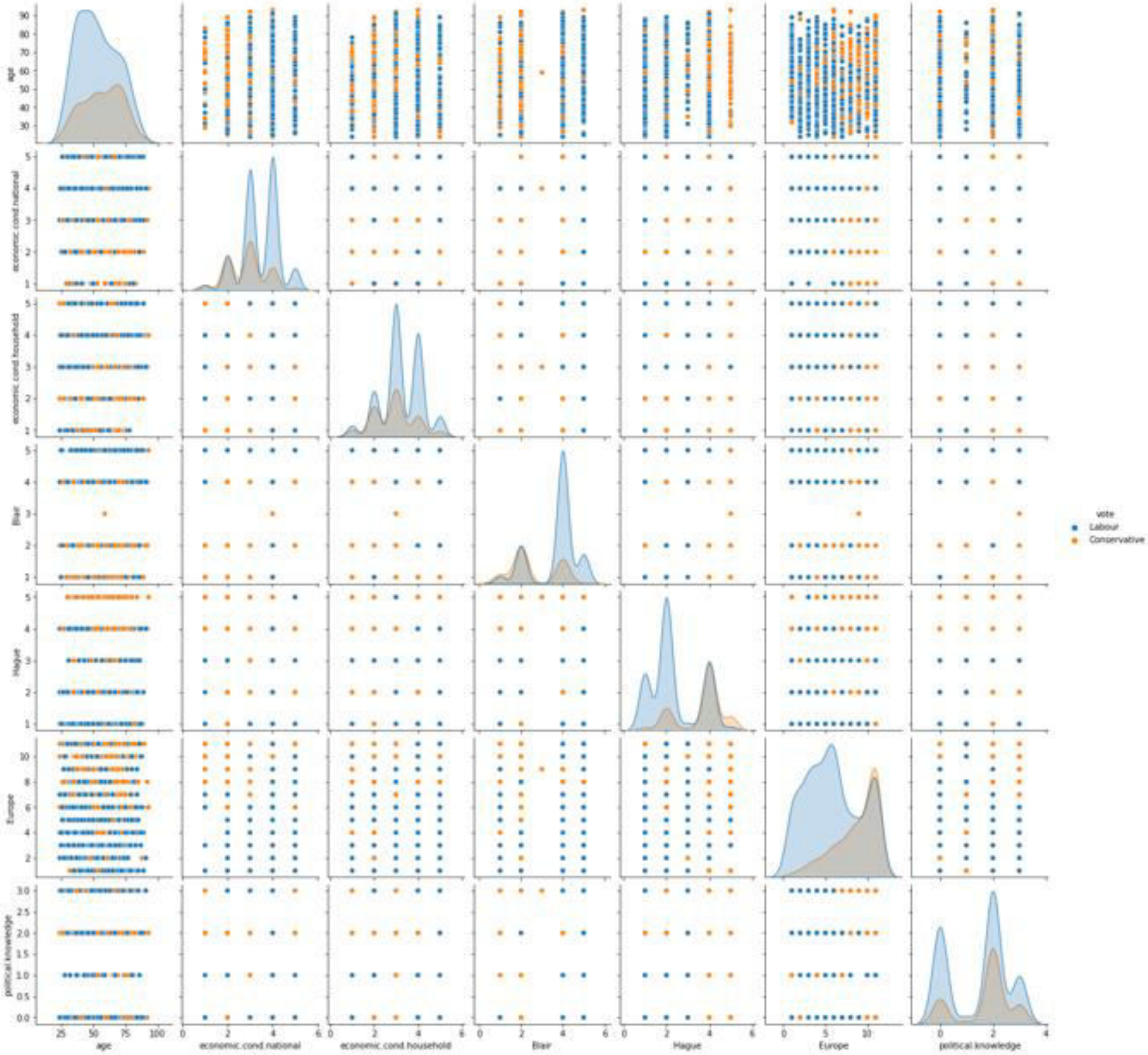


Figure 2. Pairplot for the Election Data

Inferences

- Labour party (Blair) is in blue color whereas conservative party (Hague) is in orange color.
- By looking at the age diagonal graph, majority of the people willing to vote to Labour party and it is clearly visible in the graphs because the blue is dominating over orange.
- For the Blair attribute, there is a sharp spike in the high rating whereas for Hague attribute, there is a sharp spike in the lower side of the rating.

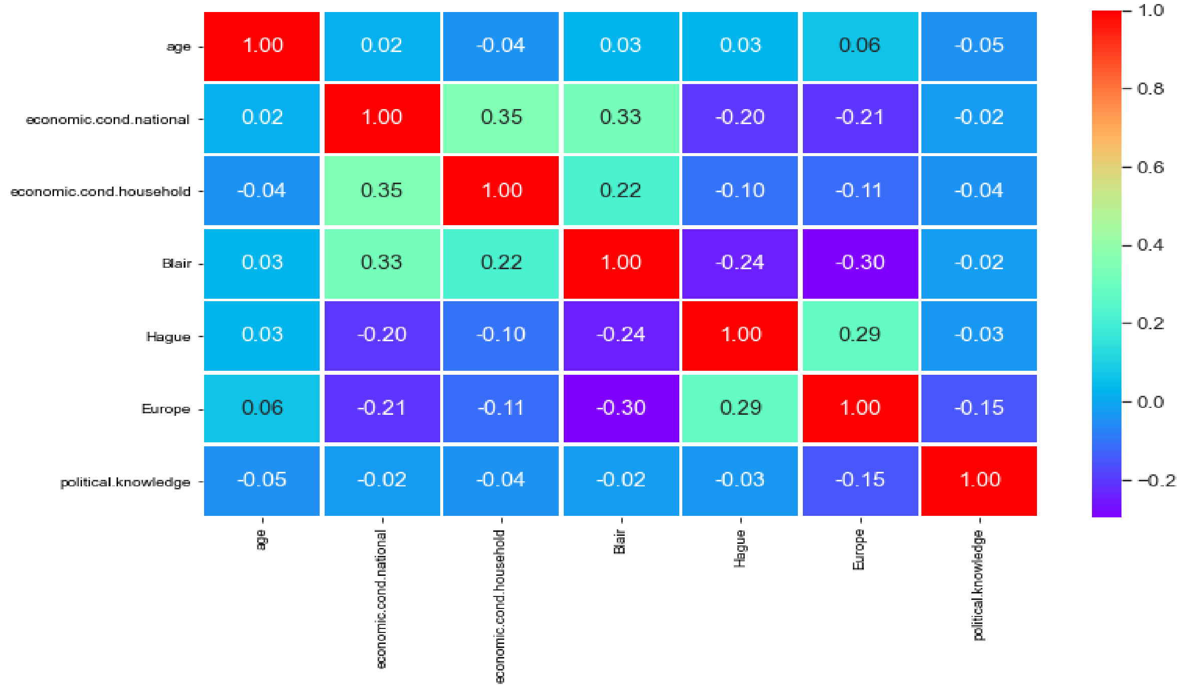


Figure 3. HeatMap for the Election Data

Inferences

There is a slight correlation between the economic.cond.national and economic.cond.household columns. Otherwise, there are weak correlations among others.

Categorical plots

Vote

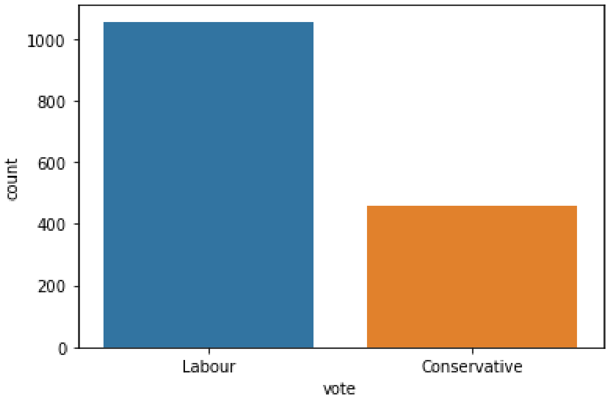


Figure 4. Bar Graph for Vote

People willing to vote to Labour party is high.

economic.cond.national and economic.cond.household Vs Vote

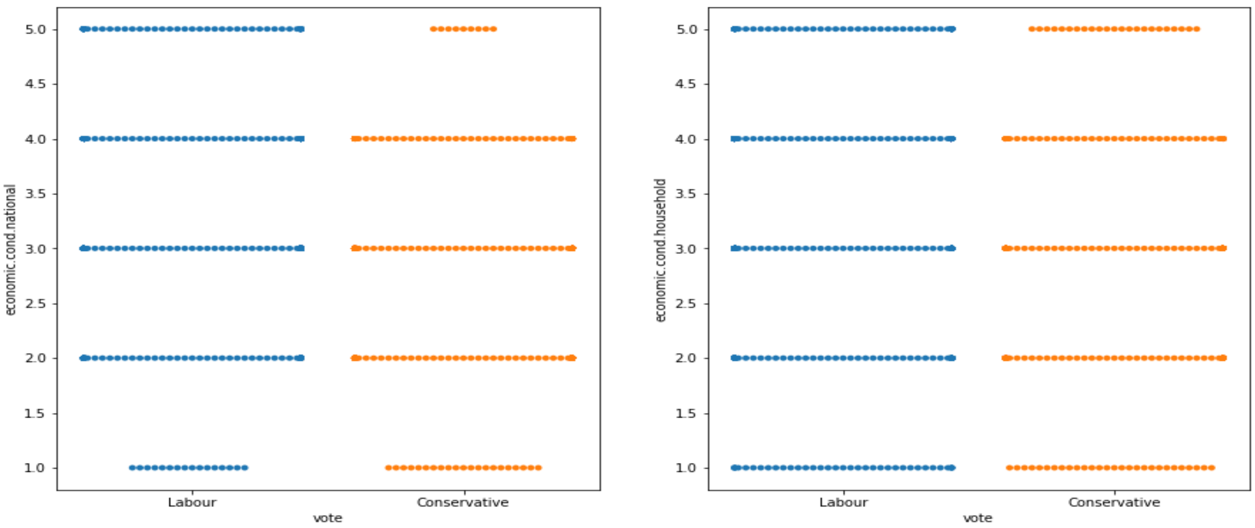


Figure 5. Swarm plot for Economic conditions of nation and household

- For the current national economic conditions Labour party has more number of 5 ratings whereas for the conservative party, the rating 1 is slightly higher compared to Labour.
- For the current household economic conditions, both the parties has got the similar ratings from 1 to 4 scale, however for the rating 5, the labour party is slightly high compared to conservative party.

Europe/Political knowledge Vs Vote

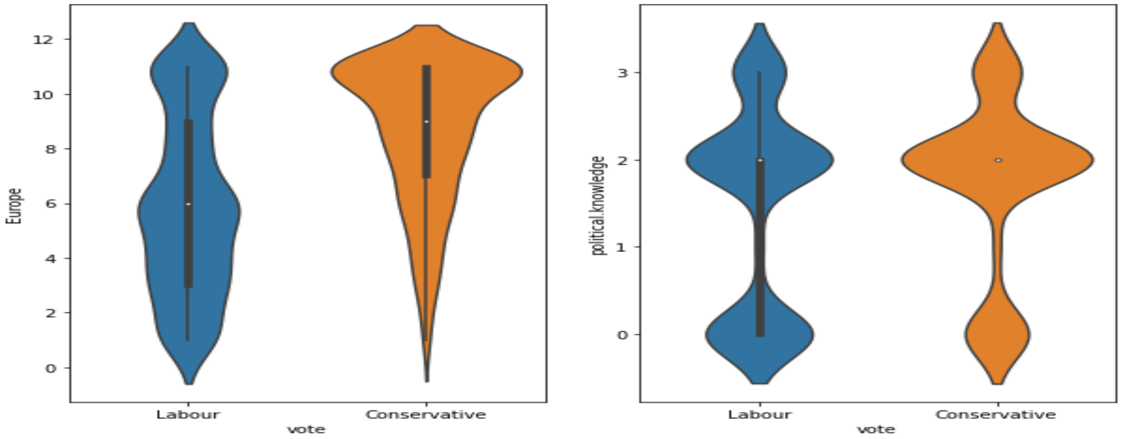


Figure 6. Violin plots for Europe/Political knowledge Vs Vote

Based on the respondents' measures towards the European Integration, Labour party is consistent on the European Integration, where as Conservative party has got the surprised ratings of either low or high rating distribution.

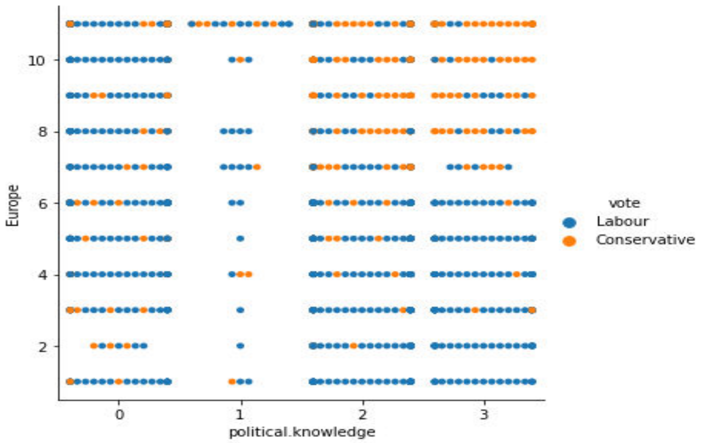


Figure 7. Categorical for Europe/Political knowledge Vs Vote

For the political knowledge, Conservative party has got more high number of ratings compared to Labour party.

Outliers

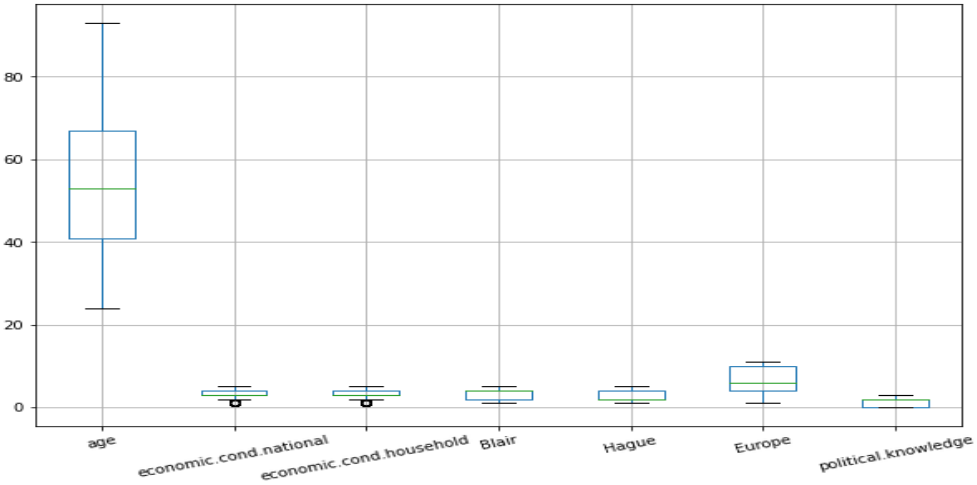
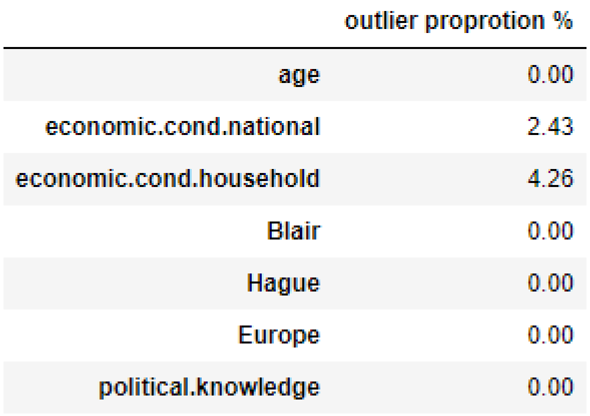


Figure 8. Boxplot for all predictors

There are very minimal outliers present in the economic conditions of nation and household columns.



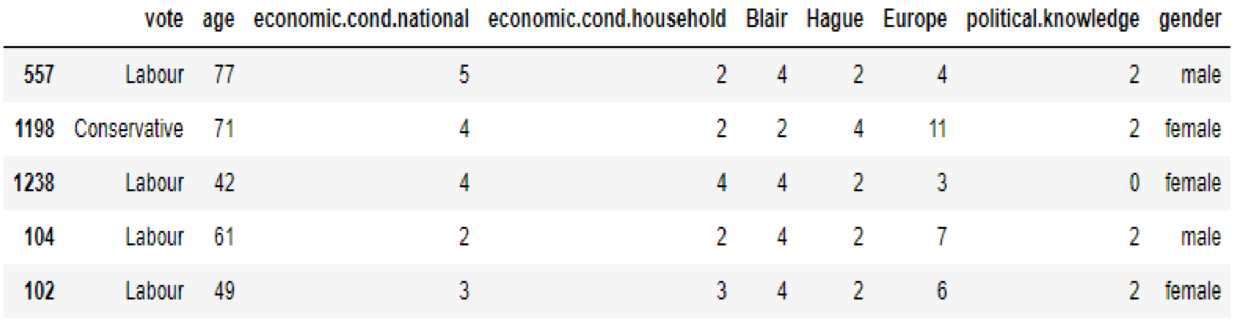
	outlier proprotion %
age	0.00
economic.cond.national	2.43
economic.cond.household	4.26
Blair	0.00
Hague	0.00
Europe	0.00
political.knowledge	0.00

Table 8. Outlier proportions of Election Data

No outlier treatment is required due to very less proportion on the outliers. Ideally, it is good to build the model without any data tampering for this business problem.

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.

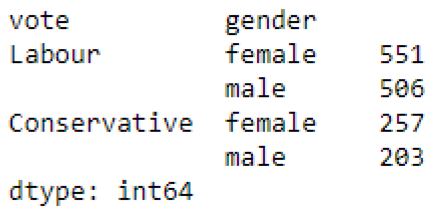
Sample records of the dataset before encoding



	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
557	Labour	77	5	2	4	2	4	2	male
1198	Conservative	71	4	2	2	4	11	2	female
1238	Labour	42	4	4	4	2	3	0	female
104	Labour	61	2	2	4	2	7	2	male
102	Labour	49	3	3	4	2	6	2	female

Table 9. Sample records before encoding

For the categorical columns ‘Vote’ and ‘gender’, we can use the categorical data encoding technique **pd.Categorical(),** since the value counts are Labour/Conservative and Male/Female.



```
vote          gender
Labour        female    551
              male      506
Conservative  female    257
              male      203
dtype: int64
```

Table 10. Value counts for categorical variables

Except age column, all the other continuous columns are in same scale.

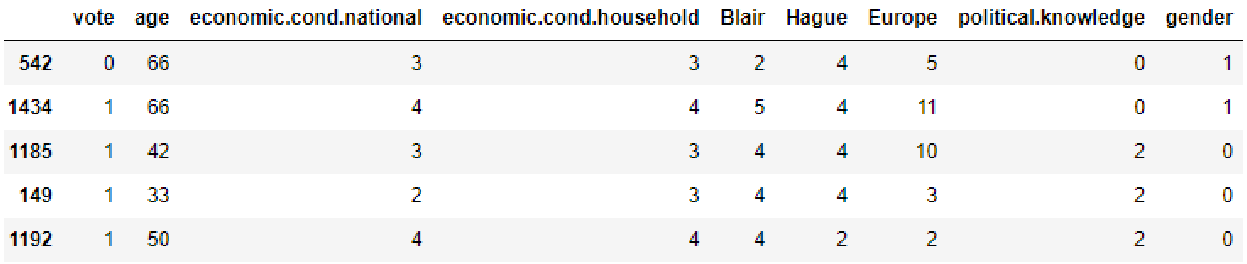
For the age column, we can use binning technique to group the age categories as follows

Age	**Category**
20-30	young
30-40	adults
40-60	mid-age
60-93	seniors

Table 11. Grouping Age

However, we will build the model without binning for now and with binning later to see any improvements in the models performance.

Sample records of the dataset after encoding

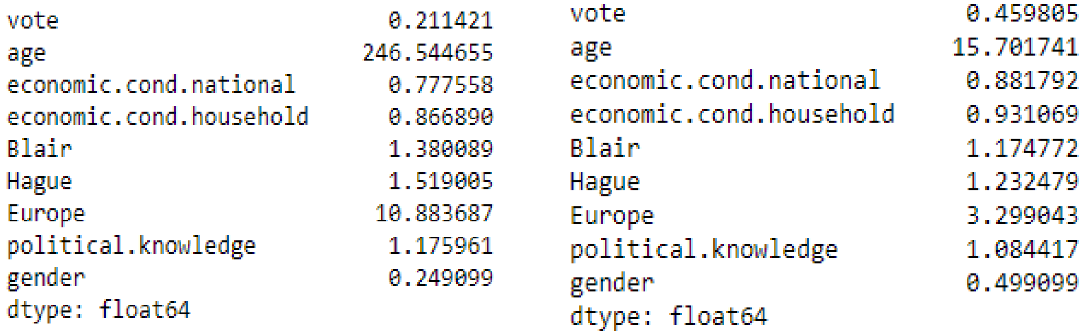


	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
542	0	66	3	3	2	4	5	0	1
1434	1	66	4	4	5	4	11	0	1
1185	1	42	3	3	4	4	10	2	0
149	1	33	2	3	4	4	3	2	0
1192	1	50	4	4	4	2	2	2	0

Table 12. Sample records after encoding

Now, we could see all the columns are numeric in nature and it’s ready for the model building process.

Variance and Standard Deviation of the dataset(Scaling or not)



```
vote                          0.211421
age                         246.544655
economic.cond.national        0.777558
economic.cond.household       0.866890
Blair                         1.380089
Hague                         1.519005
Europe                       10.883687
political.knowledge           1.175961
gender                        0.249099
dtype: float64
```

```
vote                        0.459805
age                        15.701741
economic.cond.national      0.881792
economic.cond.household     0.931069
Blair                       1.174772
Hague                       1.232479
Europe                      3.299043
political.knowledge         1.084417
gender                      0.499099
dtype: float64
```

Table 13. Variance and Standard Deviation of the Dataset

Variance helps to find the distribution of data in a population from a mean, and standard deviation also helps to know the distribution of data in population, but standard deviation gives more clarity about the deviation of data from a mean. Except for the classification algorithm like KNN, which works purely on distance based we will scale the data while building the KNN model. Otherwise, we are good to build the other classification models without scaling because except age variable, all the columns are in same scales. Even for the age variable, we have the binning technique handy to bring the age column to the similar scales like other variables in the dataframe.

Data Split

Using the sklearn package, we import **train_test_split()** function. Split the dataset, one for training the model and another one for test the model (unseen data by the model).

Throughout the problem, we build and run the models with the training data ratio of 70% and test set ratio of 30% and random state=1 to be consistent across the results. Using the '**stratify'** option; will split equally among the train and test records according to the target classification variable (vote).

Here, we assign independent columns to variable X and target column (vote) to variable y.

- Train set contains 1061 records and 8 columns.
- Test set contains 456 records and 8 columns.

Target variable percentage for train and test

	Train Labels	Test Labels
Ratio	0 0.302632 1 0.697368	0 0.302632 1 0.697368

Table 14. Vote variable percentage between train and test set

Ratio of split between train dataset and test dataset is nominal.

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 behind the selection of different values for the parameters involved in each model. Calculate Train and Test Accuracies for each model. Comment on the validness of models (over fitting or under fitting)

Logistic Regression Model

It is a Machine Learning method that is used to solve classification issues and it uses the' Sigmoid' function to classify the records.

Using the sklearn package, we import the **LogisticRegression()** function and fit the models with random_state=1.

```
log_model=LogisticRegression(random_state=1)
```

To find out the optimal hyper parameters for Logistic Model training set

After trying with different values inside the Grid Search function imported from sklearn package, the best estimators for the model is

```
LogisticRegression(max_iter=50, penalty='l1', random_state=1,
                   solver='liblinear', tol=1e-06)
```

max_iter - Maximum number of iterations taken for the solvers to converge.

n_jobs- Number of CPU cores used when parallelizing over classes, this is ignored for the liblinear solver.

solver - liblinear is preferred for the small dataset. Supports both l1 and l2 regularization.

tol - Tolerance for stopping criteria.

Let’s fit the model with the best parameters .This is the point, logistic regression model calculates the weights of each classification, and whichever the features discriminate the classes well gets more weightage and finds the best sigmoid curve using the logloss (or) cross entropy function internally.

Logistic Regression Model Accuracy

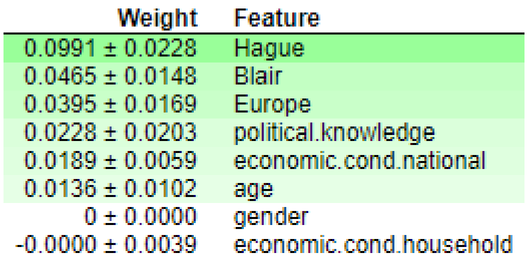
Accuracies for logistic regression model for the training and test set as shown below

- Training Set – 82.75%
- Test Set – 85.74%

Model is neither over fitting nor under fitting. It is a good valid model; however we will evaluate the model in the latter part of the document. Please refer Logistic Regression Performance metric section.

Feature Importance

Using the **eli5** sklearn package, let’s import the **PermutationImportance()** to figure out the important features for our logistic regression model.



Weight	Feature
0.0991 ± 0.0228	Hague
0.0465 ± 0.0148	Blair
0.0395 ± 0.0169	Europe
0.0228 ± 0.0203	political.knowledge
0.0189 ± 0.0059	economic.cond.national
0.0136 ± 0.0102	age
0 ± 0.0000	gender
-0.0000 ± 0.0039	economic.cond.household

Table 15. Feature Importance for logistic regression model

Feature Importance Interpretations

- The values at the top of the table are the most important features in our model, while those at the bottom matter least.
- The first number in each row indicates how much model performance decreased with random shuffling, using the same performance metric as the model (in this case, log_test_acc score).

- The number after the ± measures how performance varied from one-reshuffling to the next, i.e., degree of randomness across multiple shuffles.
- Negative values for permutation importance indicate that the predictions on the shuffled (or noisy) data are more accurate than the real data. This means that the feature does not contribute much to predictions (importance close to 0), but random chance caused the predictions on shuffled data to be more accurate.

In our example, the top 3 features are Hague, Blair and Europe, while the 3 least significant are age, gender and economic.cond.household

Linear Discriminant Analysis (LDA)

LDA is a dimensionality reduction technique that is commonly used for supervised classification problems. It is used for modeling differences in groups i.e. separating two or more classes. It works well with the small datasets.

Using the sklearn package, we import the **LinearDiscriminantAnalysis ()** function.

```
lda_model = LinearDiscriminantAnalysis()
```

<u>To find out the optimal hyper parameters for LDA Model training set</u>

After trying with different values inside the Grid Search function imported from sklearn package, the best estimators for the model is

```
LinearDiscriminantAnalysis(solver='lsqr')
```

There is not much scope to tune the model.

LDA divided the dataset into 2 groups (0 and 1), once it is separated out, now looks at the X variables and uses the method of unsupervised learning (PCA) to try and find the structure of X's. LDA model maximizes the between class variances (*like ANOVA technique*) and minimizes the within class variance (*like PCA*). It uses the Bayes' theorem to estimate the probabilities for the every new input, the class which has the highest probability is considered as the output class.

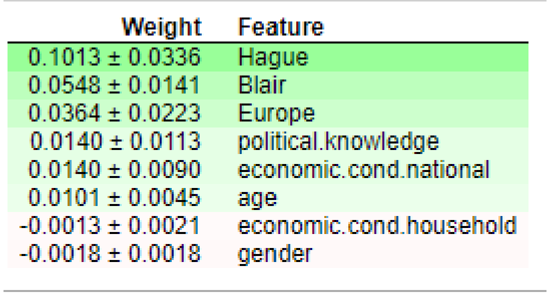
Let’s fit the model with the best parameters.

LDA Model Accuracy

Accuracies for logistic regression model for the training and test set as shown below

- Training Set – 82.28%
- Test Set – 85.30%

Model is neither over fitting nor under fitting. It is a good valid model; however we will evaluate the model in the latter part of the document. Please refer LDA Performance metric section.



Weight	Feature
0.1013 ± 0.0336	Hague
0.0548 ± 0.0141	Blair
0.0364 ± 0.0223	Europe
0.0140 ± 0.0113	political.knowledge
0.0140 ± 0.0090	economic.cond.national
0.0101 ± 0.0045	age
-0.0013 ± 0.0021	economic.cond.household
-0.0018 ± 0.0018	gender

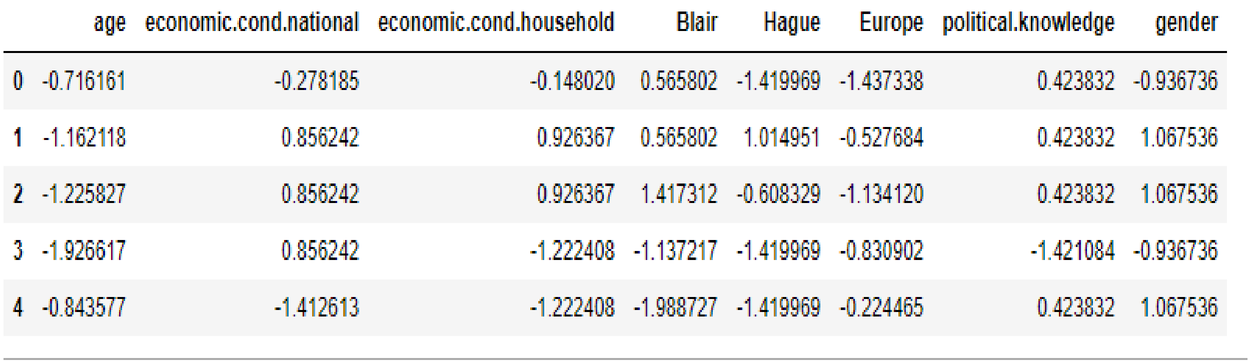
Table 16. Feature Importance for LDA model

Refer Feature Importance interpretation section for detail explanation.

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 behind the selection of different values for the parameters involved in each model. Calculate Train and Test Accuracies for each model. Comment on the validness of models (over fitting or under fitting)

KNN Model

The abbreviation KNN stands for “K-Nearest Neighbour”. It is a supervised machine learning algorithm. The algorithm can be used to solve both classification and regression problem statements. The number of nearest neighbours to a new unknown variable that has to be predicted or classified is denoted by the symbol 'K'. It calculates the ‘euclidean’ distance of K number of neighbours. Among these k neighbors, count the number of the data points in each category. Assign the new data points to that category for which the number of the neighbor is maximum. Since this algorithm works with the distance based, let’s scale the data first.



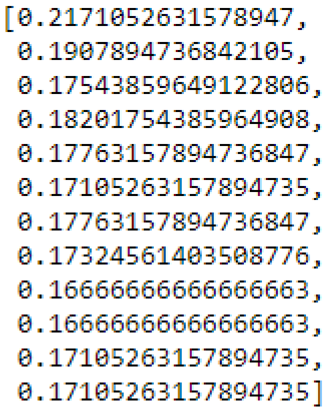
	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	-0.716161	-0.278185	-0.148020	0.565802	-1.419969	-1.437338	0.423832	-0.936736
1	-1.162118	0.856242	0.926367	0.565802	1.014951	-0.527684	0.423832	1.067536
2	-1.225827	0.856242	0.926367	1.417312	-0.608329	-1.134120	0.423832	1.067536
3	-1.926617	0.856242	-1.222408	-1.137217	-1.419969	-0.830902	-1.421084	-0.936736
4	-0.843577	-1.412613	-1.222408	-1.988727	-1.419969	-0.224465	0.423832	1.067536

Table 17. Sample records after scaling

Using the sklearn package, we import the `KNN_model=KNeighborsClassifier()` function.

<u>To find out the optimal hyper parameters for KNN Model</u>

Let's run the model with the odd number K values ranging from 1 to 25 with the step value of 2. We get the better model by finding out the minimal misclassification error.



```
[0.2171052631578947,
 0.1907894736842105,
 0.17543859649122806,
 0.18201754385964908,
 0.17763157894736847,
 0.17105263157894735,
 0.17763157894736847,
 0.17324561403508776,
 0.16666666666666663,
 0.16666666666666663,
 0.17105263157894735,
 0.17105263157894735]
```

Figure 9.Misclassification Errors

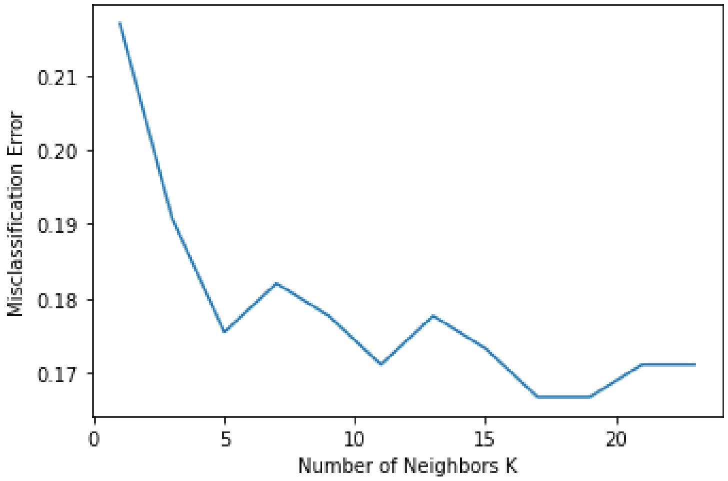


Figure 10.K Values Vs Misclassification Errors

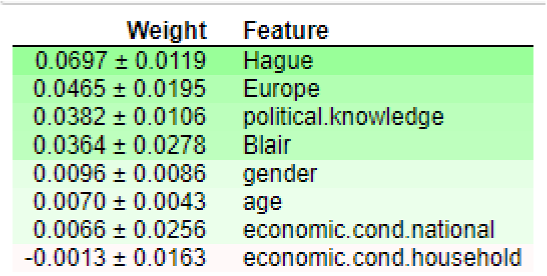
For the K value either 17 or 19, the misclassification error is straightened out. So, we will consider the optimal K value as 17 to fit the model.

KNN Model Accuracy

Accuracies KNN classification model for the training and test set as shown below

- Training Set – 100.00%
- Test Set – 82.68%

KNN Model is over fitting here. Because, the model's error on the training data is 'nil' and the test data is high. It is not a good valid model; however we will evaluate the model in the latter part of the document. Please refer KNN Performance metric section.



Weight	Feature
0.0697 ± 0.0119	Hague
0.0465 ± 0.0195	Europe
0.0382 ± 0.0106	political.knowledge
0.0364 ± 0.0278	Blair
0.0096 ± 0.0086	gender
0.0070 ± 0.0043	age
0.0066 ± 0.0256	economic.cond.national
-0.0013 ± 0.0163	economic.cond.household

Table 18. Feature Importance for KNN model

Refer Feature Importance interpretation section for detail explanation.

Naïve Bayes’ Model

Naive Bayes is a kind of classifier which uses the Bayes’ Theorem. It predicts membership probabilities for each class such as the probability that given record or data point belongs to a particular class. The class with the highest probability is considered as the most likely class.

Using the sklearn package, we import the **GaussianNB()** function.

```
NB_model = GaussianNB()
```

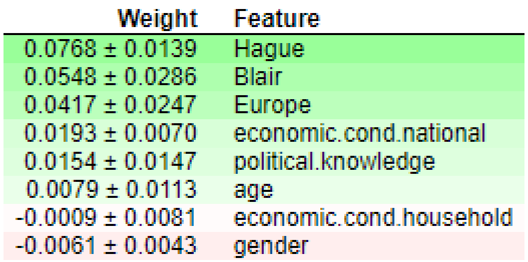
There is not much scope to tune the model. Let’s fit the model with the training records.

Naïve Bayes’ Model Accuracy

Accuracies for Naïve Bayes’ model for the training and test set as shown below

- Training Set – 83.50%
- Test Set – 82.24%

Model is neither over fitting nor under fitting. It is a good valid model; however we will evaluate the model in the latter part of the document. Please refer NB Performance metric section.



Weight	Feature
0.0768 ± 0.0139	Hague
0.0548 ± 0.0286	Blair
0.0417 ± 0.0247	Europe
0.0193 ± 0.0070	economic.cond.national
0.0154 ± 0.0147	political.knowledge
0.0079 ± 0.0113	age
-0.0009 ± 0.0081	economic.cond.household
-0.0061 ± 0.0043	gender

Table 19. Feature Importance for NB model

Refer Feature Importance interpretation section for detail explanation.

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. Define a logic behind choosing particular values for different hyper-parameters for grid search. 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.

Bagging

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it.

Using the sklearn package, we import the DecisionTreeClassifier() and BaggingClassifier() function and fit the models with random_state=1.

Let’s build the DecisionTreeClassifier() with the default gini index criterion which we will be used as the base estimator while building the ‘bagging’ classifier model.

```
dTree = DecisionTreeClassifier(criterion = 'gini', random_state=1)

bgcl_model = BaggingClassifier(base_estimator=dTree,random_state=1)
```

To find out the optimal hyper parameters for Bagging classifier Model

After trying with different values inside the Grid Search function imported from sklearn package, the best estimators for the model is

```
BaggingClassifier(base_estimator=DecisionTreeClassifier(random_state=1
),max_features=5, n_estimators=100, random_state=1)
```

n_estimators – Number of trees want to build before taking the maximum voting.

max_features- Maximum number of features is allowed to train in individual tree.

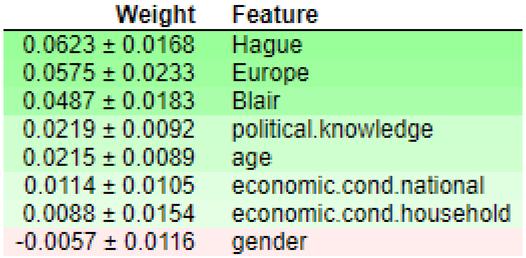
Let’s fit the model with the best parameters

Bagging Model Accuracy

Accuracies for bagging classifier model for the training and test set as shown below

- Training Set – 99.34%
- Test Set – 81.79%

Bagging classifier Model for our problem is over fitting here. Because, the model's error on the training data is almost zero and the test data is high. It is not a good valid model; however we will evaluate the model in the latter part of the document. Please refer Bagging model Performance metric section.



Weight	Feature
0.0623 ± 0.0168	Hague
0.0575 ± 0.0233	Europe
0.0487 ± 0.0183	Blair
0.0219 ± 0.0092	political.knowledge
0.0215 ± 0.0089	age
0.0114 ± 0.0105	economic.cond.national
0.0088 ± 0.0154	economic.cond.household
-0.0057 ± 0.0116	gender

Table 20. Feature Importance for Bagging model

Refer Feature Importance interpretation section for detail explanation.

Random Forest Classifier

The Random Forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

Using the sklearn package, we import the **RandomForestClassifier()** function and fit the models with random_state=1.

```
RF_model=RandomForestClassifier(random_state=1)
```

To find out the optimal hyper parameters for Random Forest classifier Model

After trying with different values inside the Grid Search function imported from sklearn package, the best estimators for the model is

```
RandomForestClassifier(n_estimators=100, max_features=7,
random_state=1)
```

n_estimators – Number of trees want to build before taking the maximum voting.

max_features- Maximum number of features is allowed to train in individual tree.

Let's fit the model with the best parameters

Random Forest Model Accuracy

Accuracies for Random Forest classifier model for the training and test set as shown below

- Training Set – 100.00%
- Test Set – 82.01%

Random Forest classifier Model for our problem is over fitting here. Because, the model's error on the training data is zero and the test data is high. It is not a good valid model; however we will evaluate the model in the latter part of the document. Please refer RF model Performance metric section.

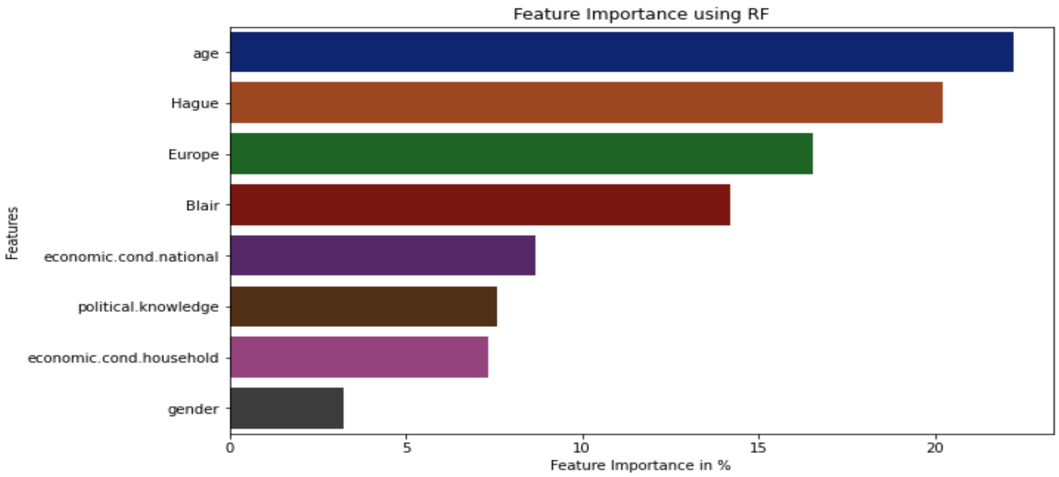
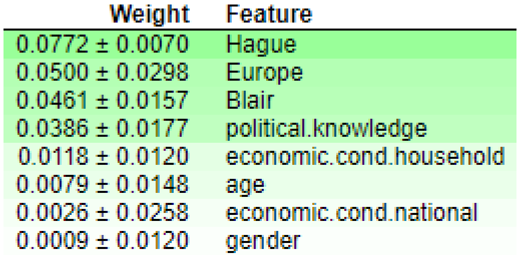


Figure 11.Feature Importance for Training set



Weight	Feature
0.0772 ± 0.0070	Hague
0.0500 ± 0.0298	Europe
0.0461 ± 0.0157	Blair
0.0386 ± 0.0177	political.knowledge
0.0118 ± 0.0120	economic.cond.household
0.0079 ± 0.0148	age
0.0026 ± 0.0258	economic.cond.national
0.0009 ± 0.0120	gender

Table 21. Feature Importance for RF model

Refer Feature Importance interpretation section for detail explanation.

Ada Boosting

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble method. The most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees with only 1 split. These trees are also called Decision Stumps. AdaBoost works by putting more weight on difficult to classify instances and less on those already handled well. When nothing works, boosting works well for most of the common business problems.

Using the sklearn package, we import the **AdaBoostClassifier()** function and fit the models with random_state=1.

```
ADB_model = AdaBoostClassifier(n_estimators=100,random_state=1)
```

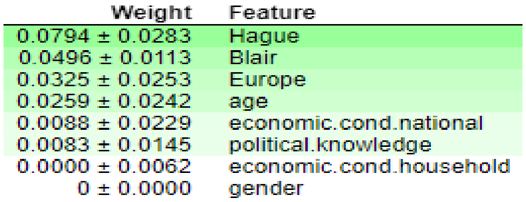
There is not much scope to tune the model. Let's fit the model with the training records.

Ada Boosting Model Accuracy

Accuracies for Ada Boosting model for the training and test set as shown below

- Training Set – 85.01%
- Test Set – 81.35%

Model is neither over fitting nor under fitting. It is a good valid model; however we will evaluate the model in the latter part of the document. Please refer Ada Boosting Performance metric section.



Weight	Feature
0.0794 ± 0.0283	Hague
0.0496 ± 0.0113	Blair
0.0325 ± 0.0253	Europe
0.0259 ± 0.0242	age
0.0088 ± 0.0229	economic.cond.national
0.0083 ± 0.0145	political.knowledge
0.0000 ± 0.0062	economic.cond.household
0 ± 0.0000	gender

Table 22. Feature Importance for Ada Boosting model

Refer Feature Importance interpretation section for detail explanation.

Gradient Boosting

In Gradient Boosting, each predictor tries to improve on its predecessor by reducing the errors. But the fascinating idea behind Gradient Boosting is that instead of fitting a predictor on the data at each iteration, it actually fits a new predictor t*o the residual errors made by the previous predictor.*

Using the sklearn package, we import the **GradientBoostingClassifier ()** function and fit the models with random_state=1.

To find out the optimal hyper parameters for Gradient Boosting Model

After trying with different values inside the Grid Search function imported from sklearn package, the best estimators for the model is

```
GradientBoostingClassifier(max_features=5,n_estimators=100,
tol=0.0001,random_state=1)
```

n_estimators The number of boosting stages to perform.

max_features- The number of features to consider when looking for the best split.

tol- Tolerance for the early stopping.

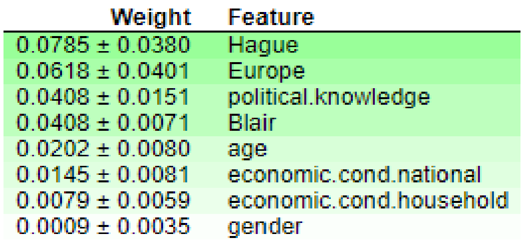
Let's fit the model with the best parameters

Gradient Boosting Model Accuracy

Accuracies for Gradient Boosting model for the training and test set as shown below

- Training Set – 89.16%
- Test Set – 83.55%

Model is neither over fitting nor under fitting. It is a good valid model; however we will evaluate the model in the latter part of the document. Please refer Gradient Boosting Performance metric section.



Weight	Feature
0.0785 ± 0.0380	Hague
0.0618 ± 0.0401	Europe
0.0408 ± 0.0151	political.knowledge
0.0408 ± 0.0071	Blair
0.0202 ± 0.0080	age
0.0145 ± 0.0081	economic.cond.national
0.0079 ± 0.0059	economic.cond.household
0.0009 ± 0.0035	gender

Table 23. Feature Importance for Ada Boosting model

Refer Feature Importance interpretation section for detail explanation.

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)

Our business problem is to analyze the recent UK elections and to predict the overall win and seats covered by the particular party. In our case, 0 refers to 'Conservative' party led by Hague and 1 refers to 'Labour' party led by Blair. Both the classification is equally important, within that 'precision' is slightly more important than recall. It is the measure of voters that we correctly identify the voters willing to vote to Labour or Conservative party out of all voters. It is the ratio between true positives and all the positives.

Precision and Recall are the two building blocks of the F1 score. The goal of the F1 score is to combine the precision and recall metrics into a single metric. At the same time, the F1 score has been designed to work well on imbalanced data.

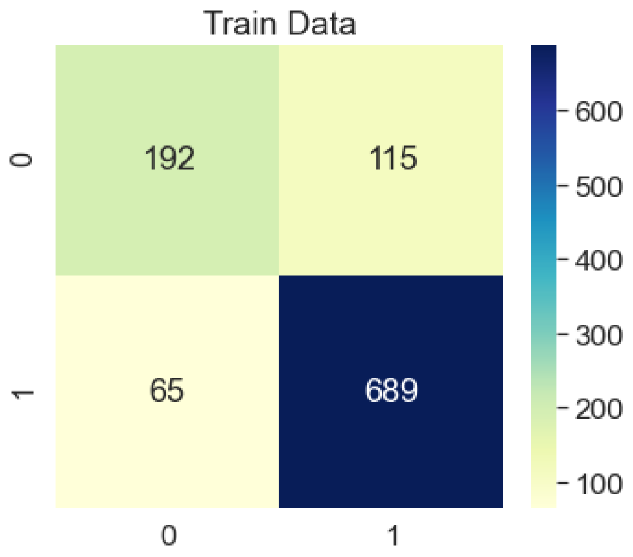
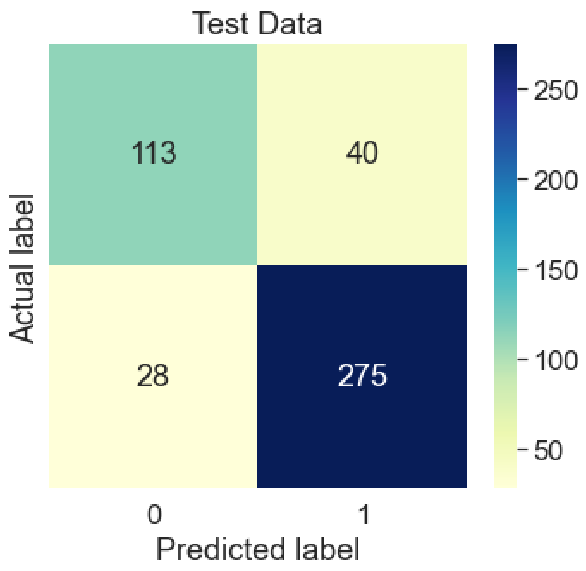
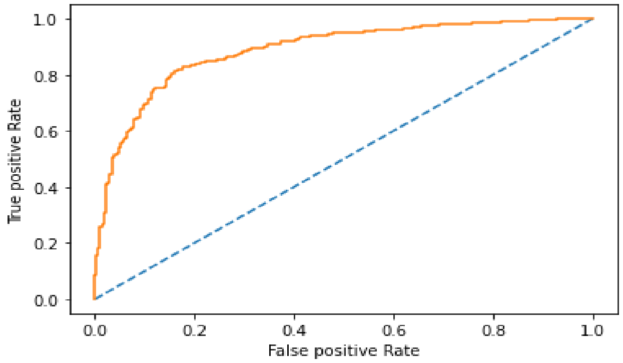
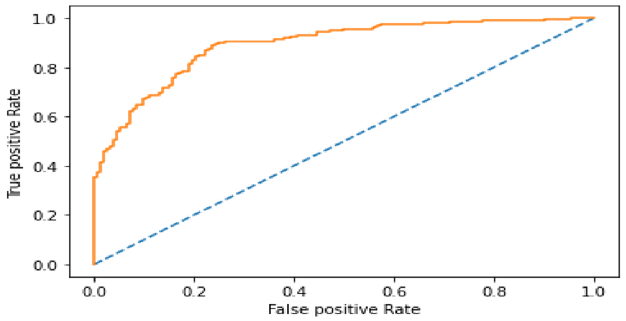
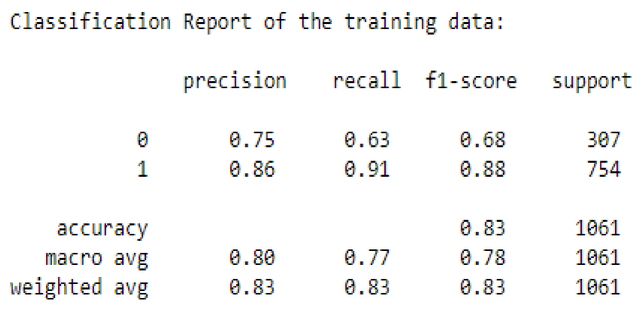
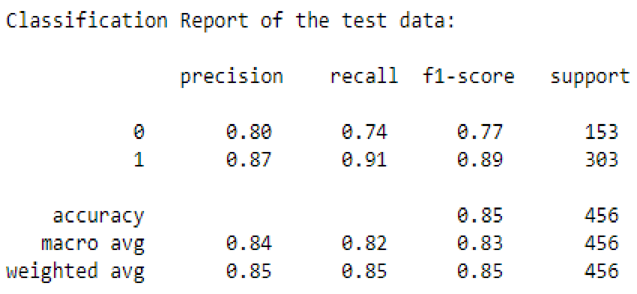
When we need to check or visualize the performance of the classification problem, we use the AUC (**Area Under The Curve**) ROC (**Receiver Operating Characteristics**) curve. It is one of the most important evaluation metrics for checking any classification model's performance. AUC - ROC curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. Always, aim for high AUC score.

We calculate Accuracy, confusion matrix, ROC Curve, AUC score and classification report for both training and test set for all the classification models built.

There are 1061 records for the training set and 456 records for the testing set.

greatlearning
Power Ahead

Logistic Regression Model Performance Metrics

Training Set	Test Set
Accuracy Score – **83%**	Accuracy Score – **85%**
Confusion Matrix  	Confusion Matrix 
ROC Curve  	ROC Curve 
ROC_AUC Score – **88.7%**	ROC_AUC Score – **89.2%**
Classification Report 	Classification Report 

```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       0.75      0.63      0.68       307
           1       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
```

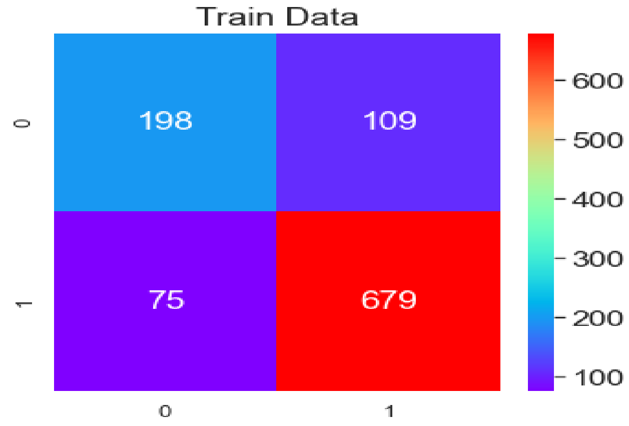
```
Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.80      0.74      0.77       153
           1       0.87      0.91      0.89       303

    accuracy                           0.85       456
   macro avg       0.84      0.82      0.83       456
weighted avg       0.85      0.85      0.85       456
```

Linear Discriminant Analysis Performance Metrics

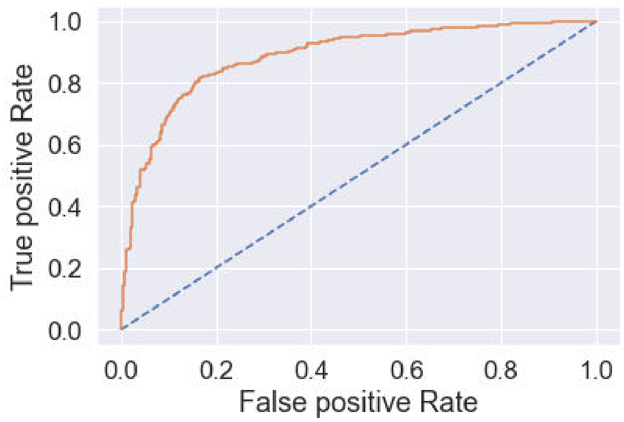
Training Set

Accuracy Score – **82.66%**

Confusion Matrix

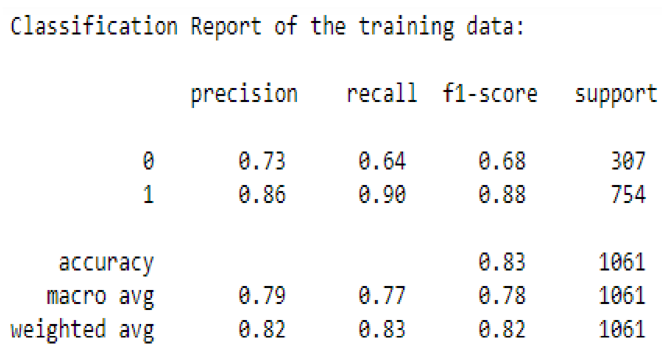


ROC Curve



ROC_AUC Score – **88.6%**

Classification Report

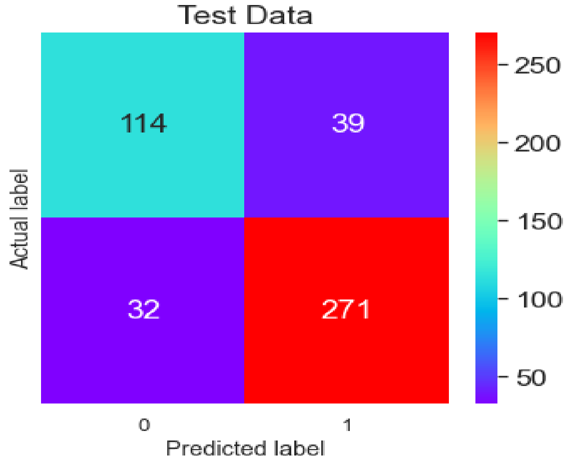


```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       0.73      0.64      0.68       307
           1       0.86      0.90      0.88       754

    accuracy                           0.83      1061
   macro avg       0.79      0.77      0.78      1061
weighted avg       0.82      0.83      0.82      1061
```

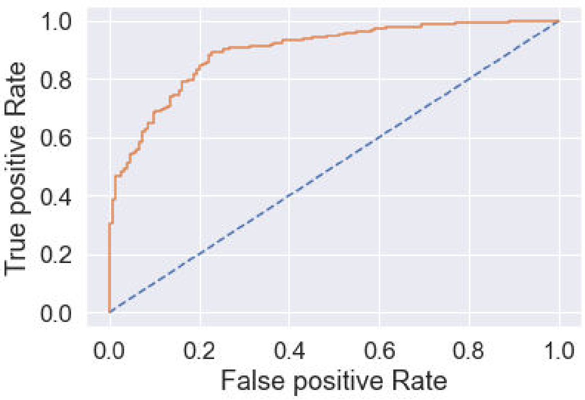
Test Set

Accuracy Score – **84.43%**

Confusion Matrix

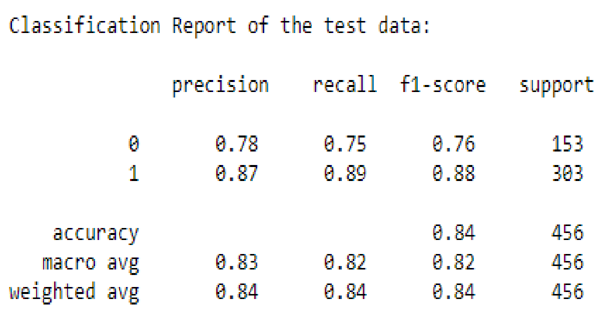


ROC Curve



ROC_AUC Score – **89.4%**

Classification Report



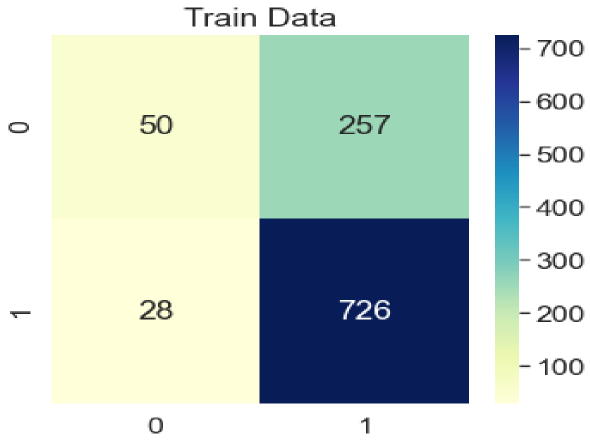
```
Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.78      0.75      0.76       153
           1       0.87      0.89      0.88       303

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

KNN Model Performance Metrics

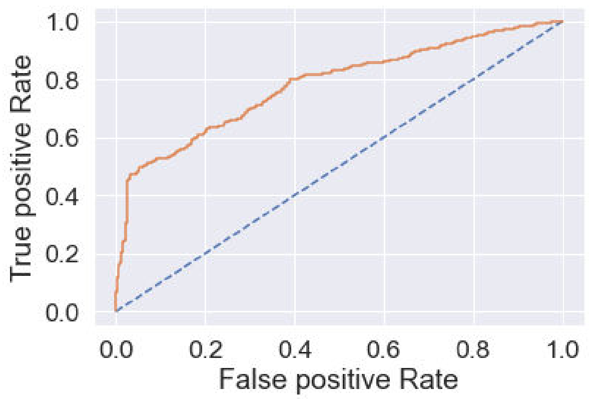
Training Set

Accuracy Score – 73.13%

Confusion Matrix

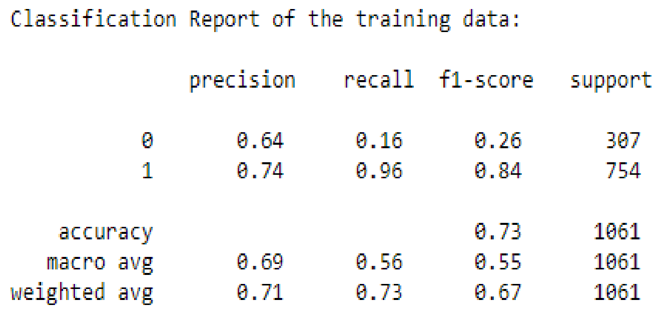


ROC Curve



ROC_AUC Score – 78.1%

Classification Report

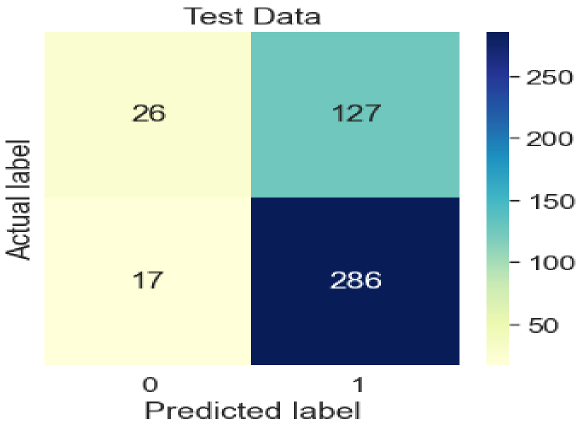


```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       0.64      0.16      0.26       307
           1       0.74      0.96      0.84       754

    accuracy                           0.73      1061
   macro avg       0.69      0.56      0.55      1061
weighted avg       0.71      0.73      0.67      1061
```

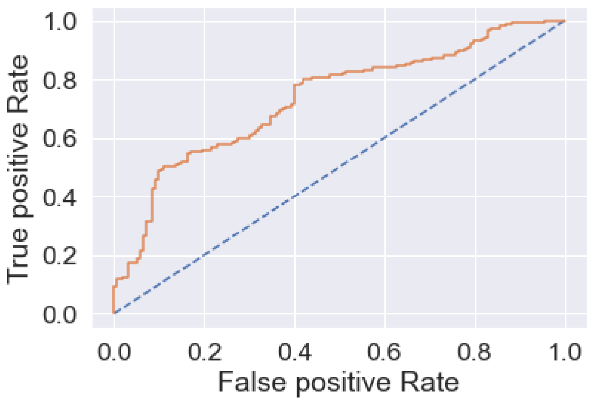
Test Set

Accuracy Score – 68.42%

Confusion Matrix

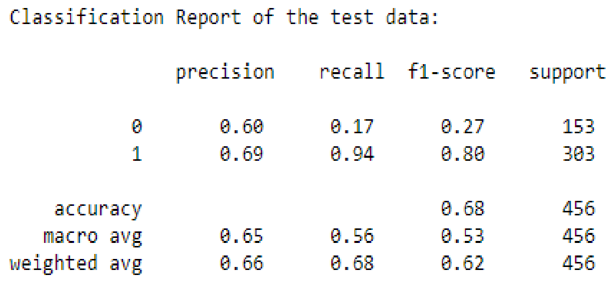


ROC Curve



ROC_AUC Score – 73.3%

Classification Report



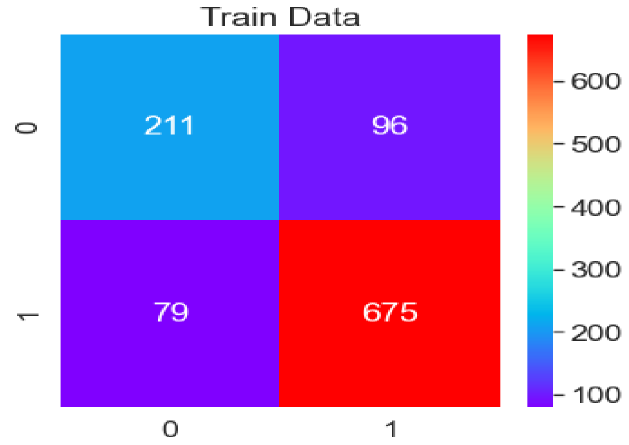
```
Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.60      0.17      0.27       153
           1       0.69      0.94      0.80       303

    accuracy                           0.68       456
   macro avg       0.65      0.56      0.53       456
weighted avg       0.66      0.68      0.62       456
```

Naïve Bayes' Performance Metrics

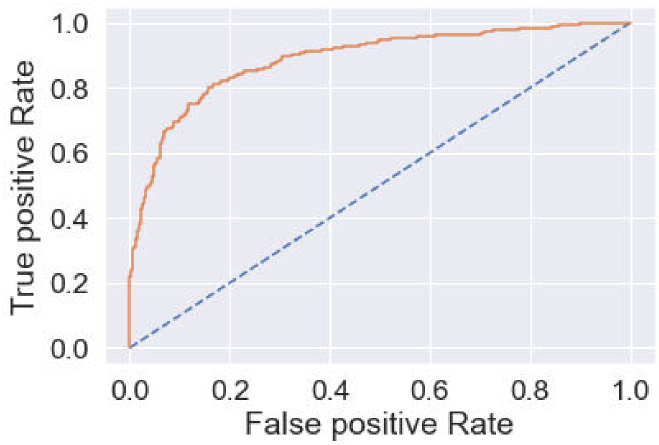
Training Set

Accuracy Score – 83.50%

Confusion Matrix

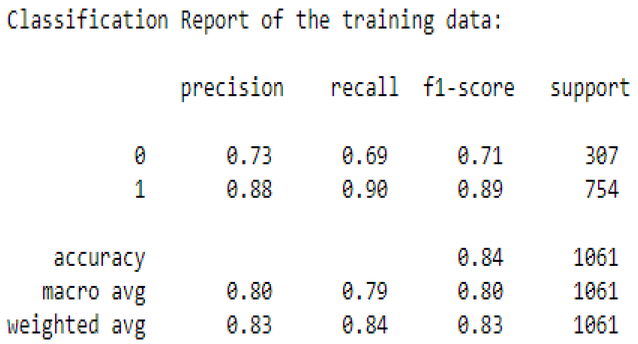


ROC Curve



ROC_AUC Score – 88.8%

Classification Report

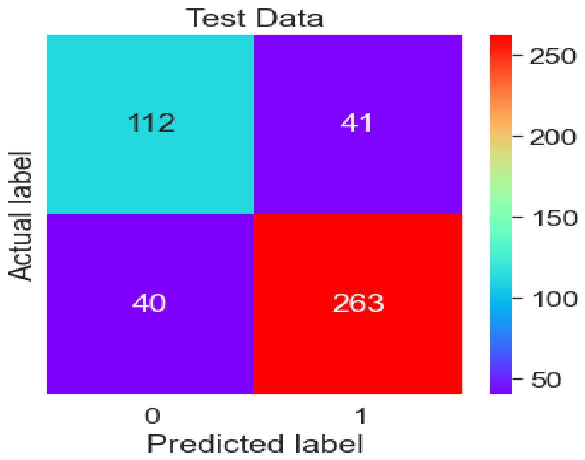


```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       0.73      0.69      0.71       307
           1       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
```

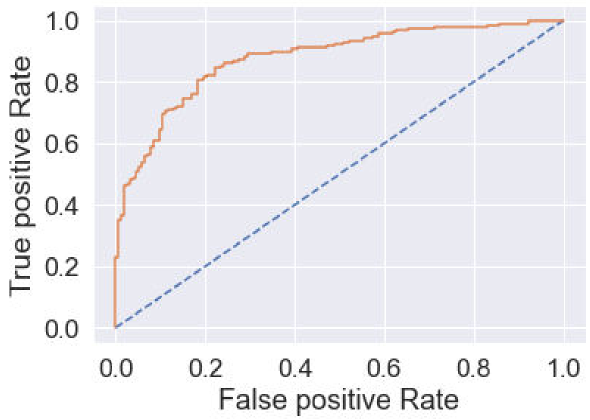
Test Set

Accuracy Score – 82.23%

Confusion Matrix

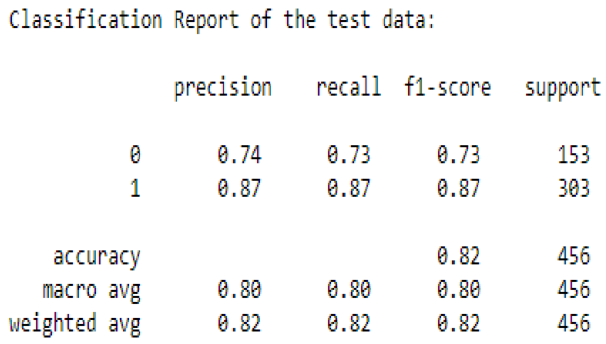


ROC Curve



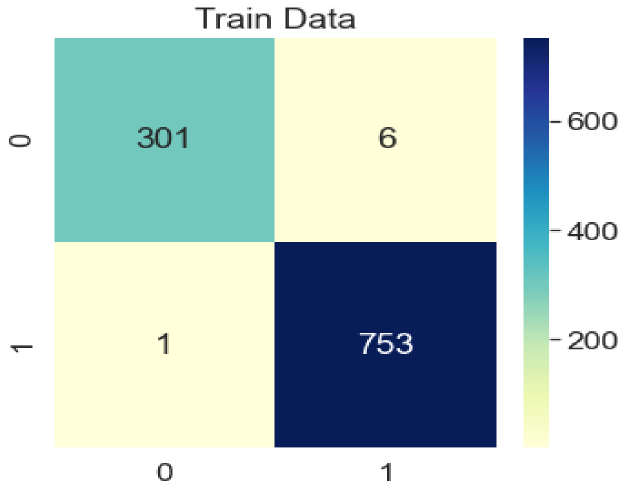
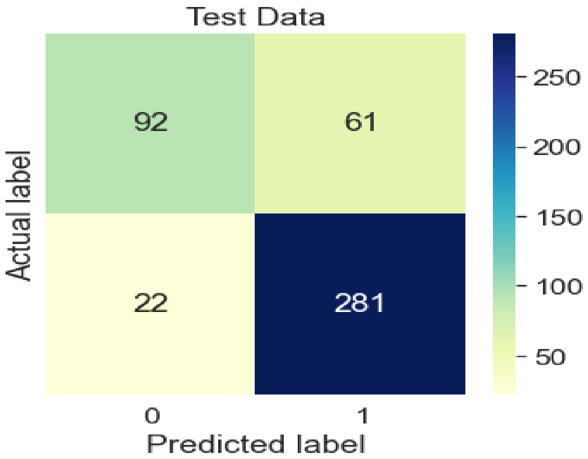
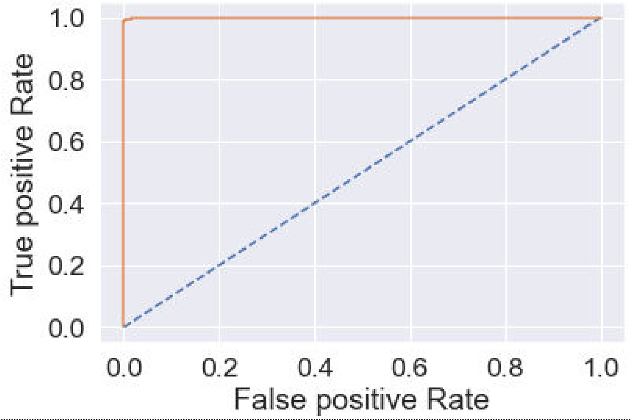
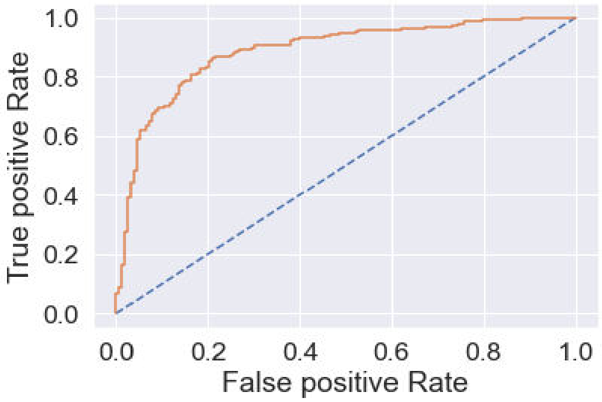
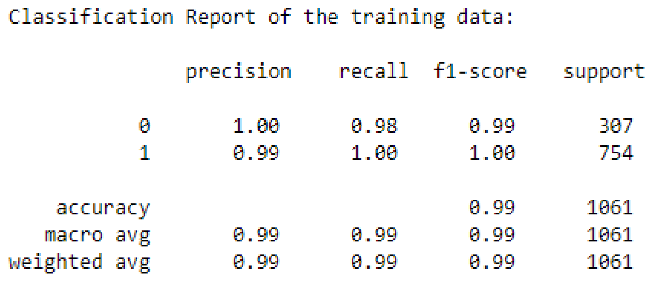
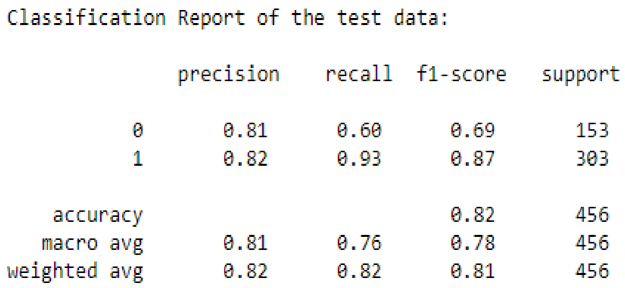
ROC_AUC Score – 87.6%

Classification Report



```
Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.74      0.73      0.73       153
           1       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
```

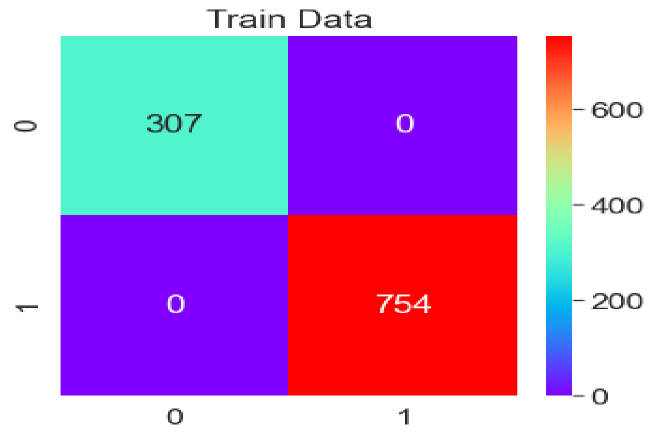
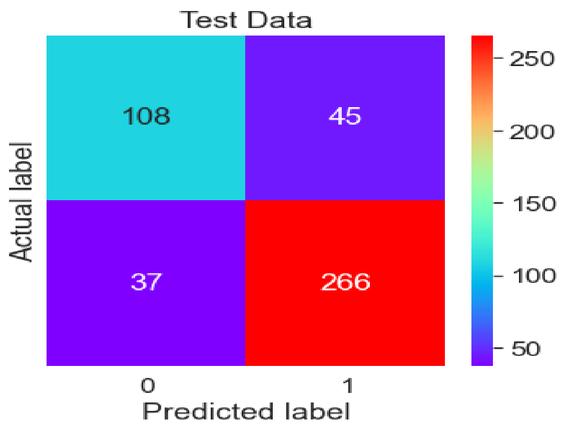
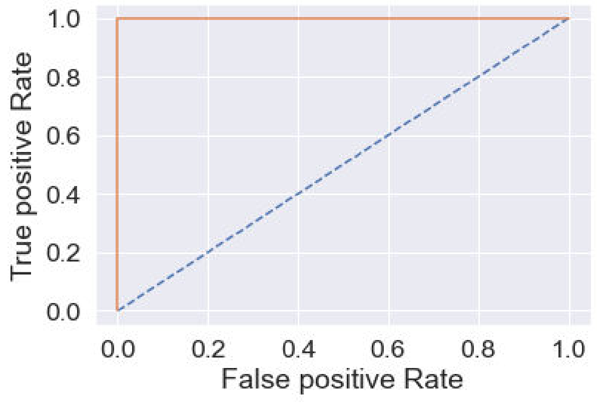
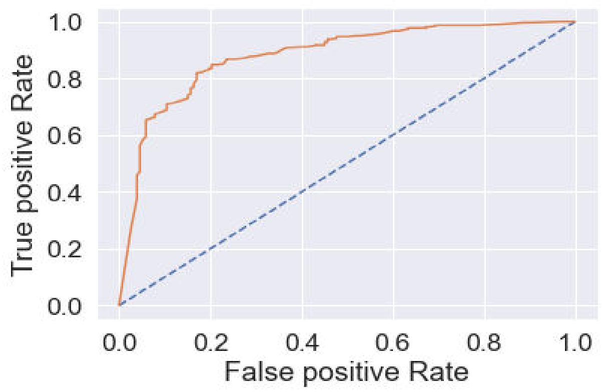
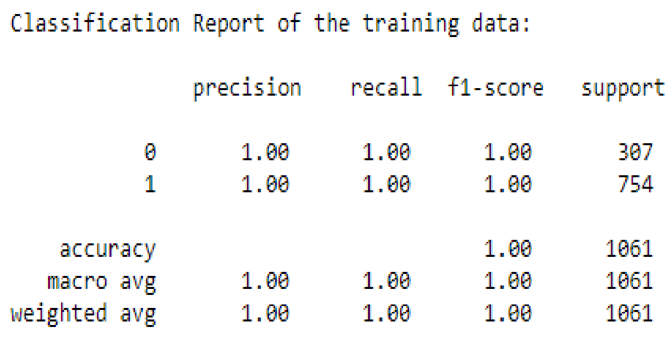
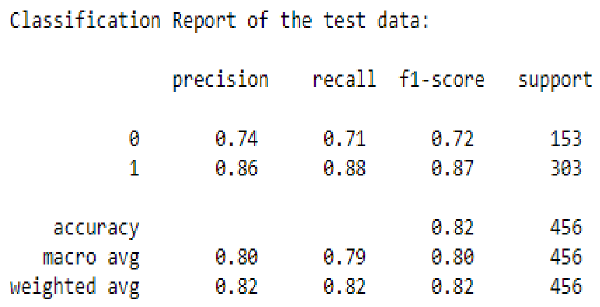
Bagging Classifier Performance Metrics

Training Set	Test Set
Accuracy Score – **99.34%**	Accuracy Score – **81.79%**
Confusion Matrix  	Confusion Matrix 
ROC Curve  	ROC Curve 
ROC_AUC Score – **100%**	ROC_AUC Score – **88.6%**
Classification Report 	Classification Report 

```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       1.00      0.98      0.99       307
           1       0.99      1.00      1.00       754

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

```
Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.81      0.60      0.69       153
           1       0.82      0.93      0.87       303

    accuracy                           0.82       456
   macro avg       0.81      0.76      0.78       456
weighted avg       0.82      0.82      0.81       456
```

Random Forest Classifier Performance Metrics

Training Set	Test Set
Accuracy Score – **100%**	Accuracy Score – **82%**
Confusion Matrix 	Confusion Matrix 
ROC Curve 	ROC Curve 
ROC_AUC Score – **100%**	ROC_AUC Score – **88.1%**
Classification Report 	Classification Report 

```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00       307
           1       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
```

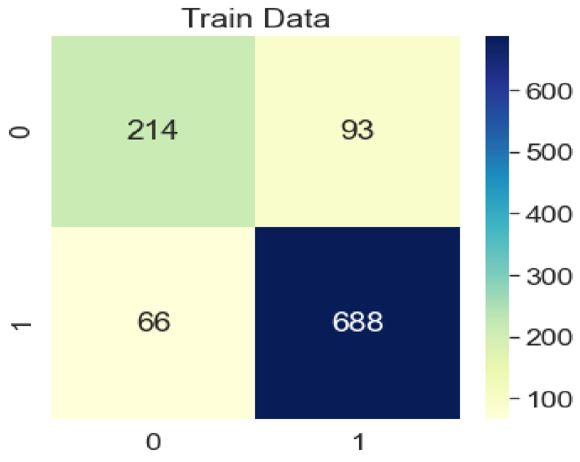
```
Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.74      0.71      0.72       153
           1       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
```

Ada Boosting Performance Metrics

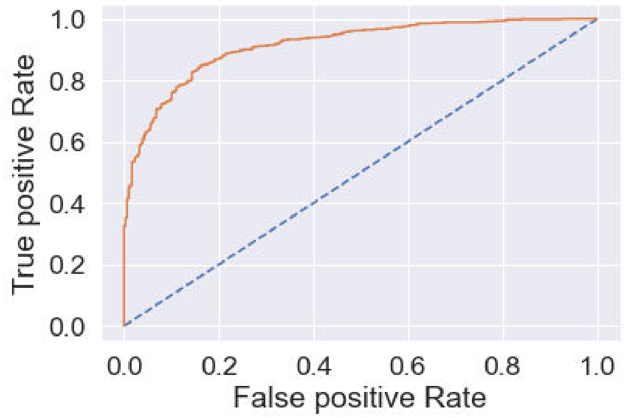
Training Set

Accuracy Score – 85%

Confusion Matrix

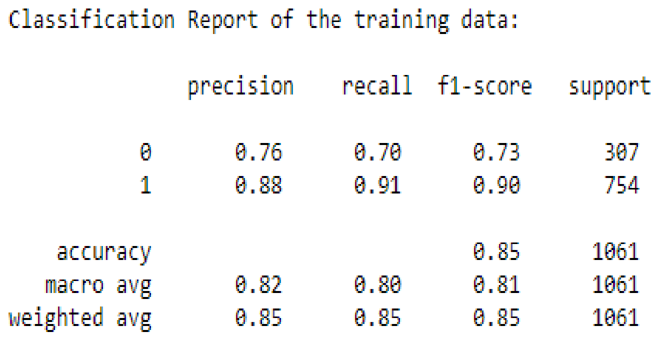


ROC Curve



ROC_AUC Score – 91.5%

Classification Report

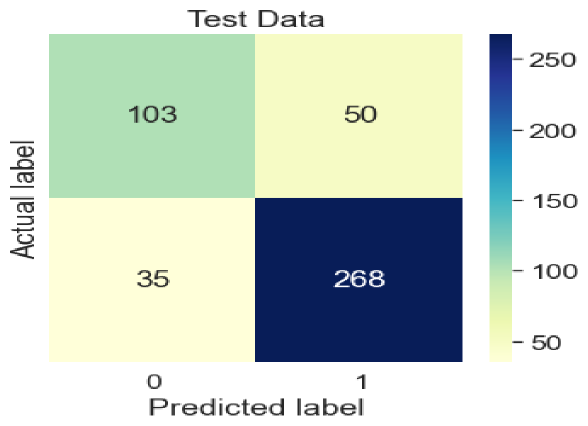


```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       0.76      0.70      0.73       307
           1       0.88      0.91      0.90       754

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

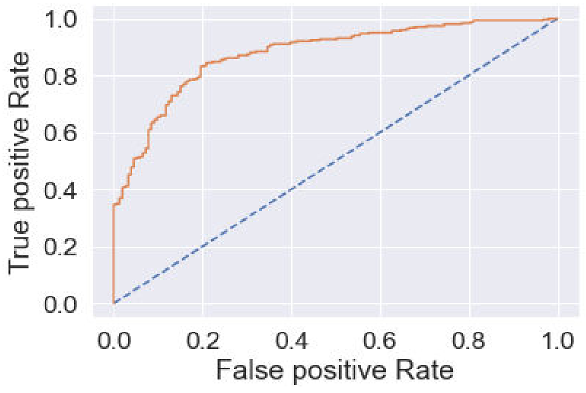
Test Set

Accuracy Score – 81.35%

Confusion Matrix

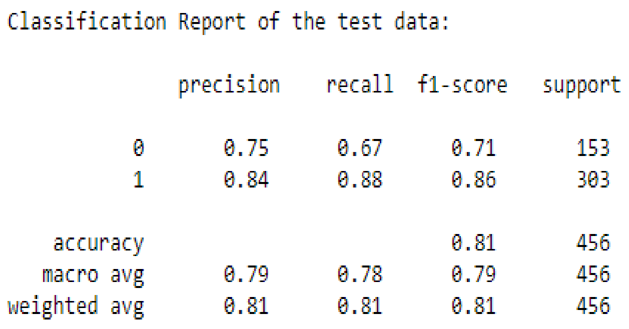


ROC Curve



ROC_AUC Score – 87.7%

Classification Report



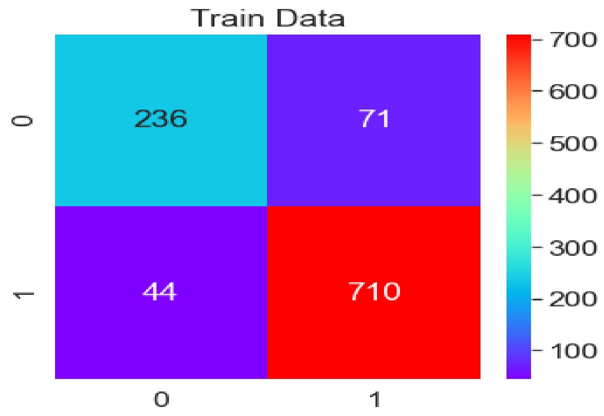
```
Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.75      0.67      0.71       153
           1       0.84      0.88      0.86       303

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

Gradient Boosting Performance Metrics

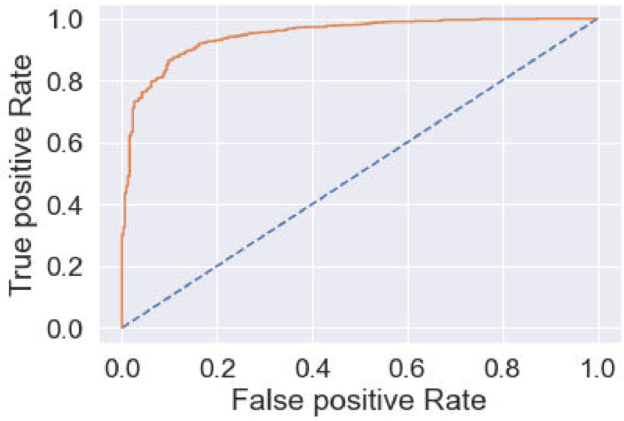
Training Set

Accuracy Score – 89.16%

Confusion Matrix

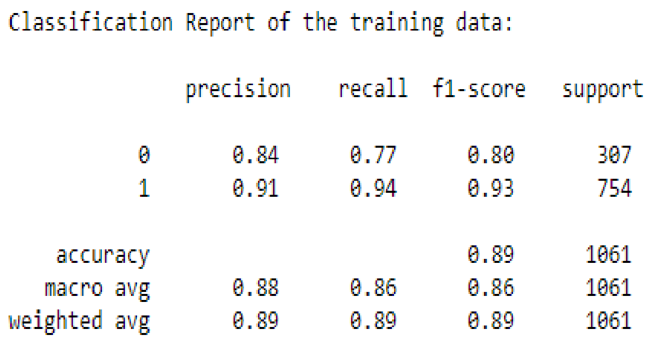


ROC Curve



ROC_AUC Score – 94.8%

Classification Report

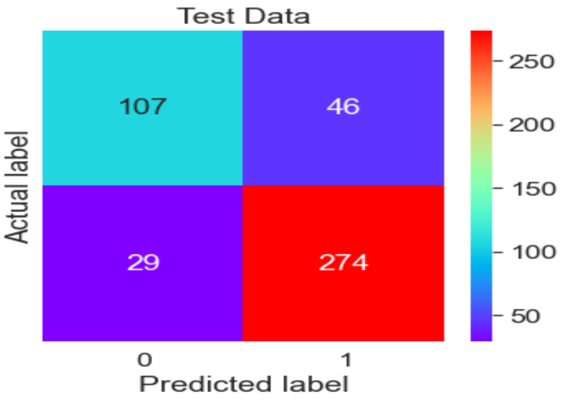


```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       0.84      0.77      0.80       307
           1       0.91      0.94      0.93       754

    accuracy                           0.89      1061
   macro avg       0.88      0.86      0.86      1061
weighted avg       0.89      0.89      0.89      1061
```

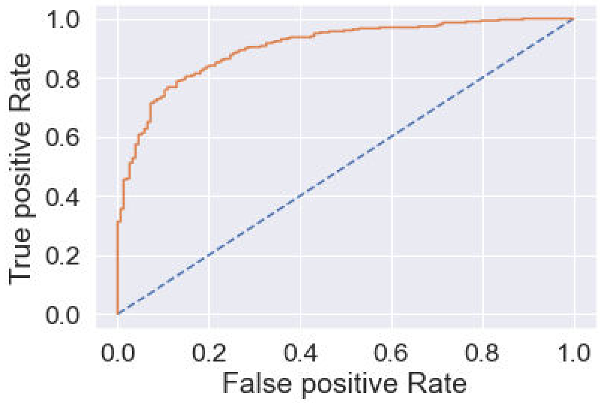
Test Set

Accuracy Score – 83.55%

Confusion Matrix

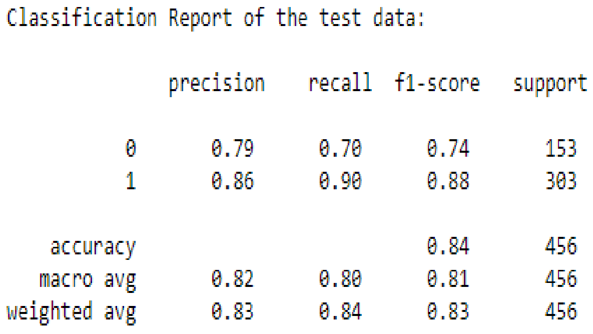


ROC Curve



ROC_AUC Score – 90.4%

Classification Report



```
Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.79      0.70      0.74       153
           1       0.86      0.90      0.88       303

    accuracy                           0.84       456
   macro avg       0.82      0.80      0.81       456
weighted avg       0.83      0.84      0.83       456
```

ROC Curve and AUC score for all the Models

Area under the curve for Logistic Regression Classification Model is 0.891930369507539
Area under the curve for LDA Classification Model is 0.8943031558057766
Area under the curve for KNN Calssification Model is 0.7334282447852628
Area under the curve for Naive Bayes Classification Model is 0.8763562630772882
Area under the curve for Bagging Classification Model is 0.8863219655298864
Area under the curve for Random Forest Classification Model is 0.8814901097952933
Area under the curve for Ada Boosting Classification Model is 0.8773808753424363
Area under the curve for Gradient Boosting Classification Model is 0.9037619448219331

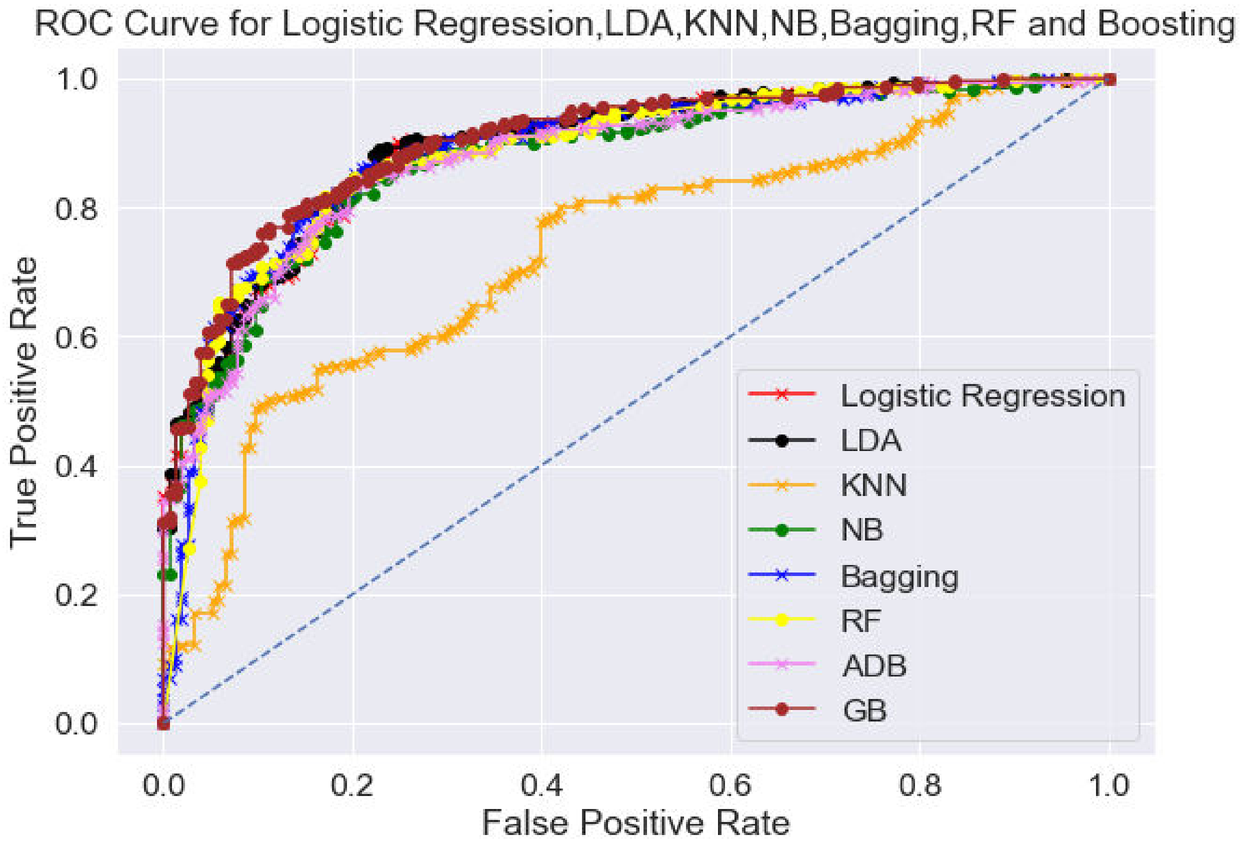
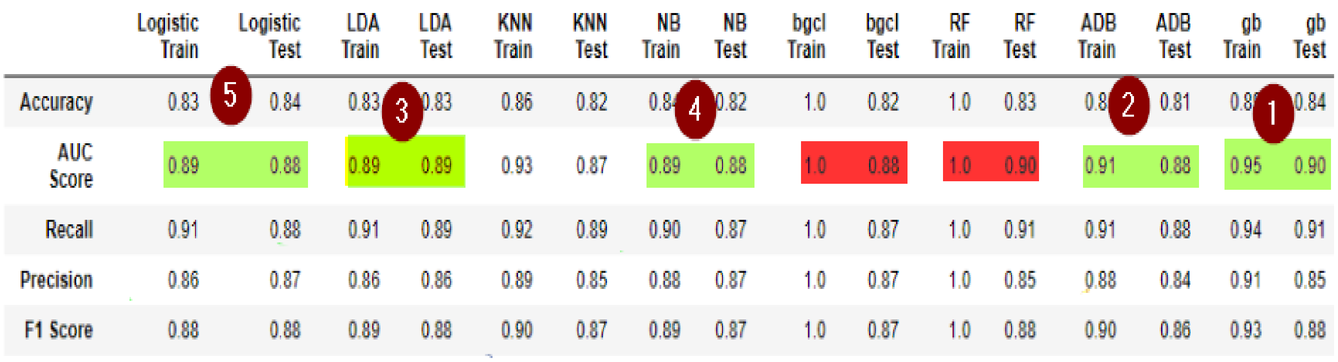


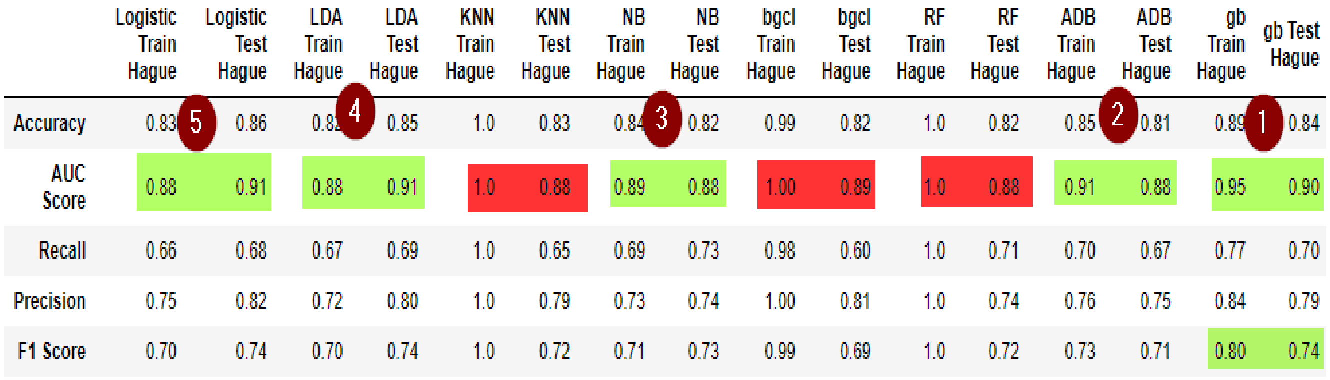
Figure 12.ROC Curve for all the Models

Comparison Summary of all Models



	Logistic Train	Logistic Test	LDA Train	LDA Test	KNN Train	KNN Test	NB Train	NB Test	bgcl Train	bgcl Test	RF Train	RF Test	ADB Train	ADB Test	gb Train	gb Test
Accuracy	0.83	0.84	0.83	0.83	0.86	0.82	0.8	0.82	1.0	0.82	1.0	0.83	0.8	0.81	0.8	0.84
AUC Score	0.89	0.88	0.89	0.89	0.93	0.87	0.89	0.88	1.0	0.88	1.0	0.90	0.91	0.88	0.95	0.90
Recall	0.91	0.88	0.91	0.89	0.92	0.89	0.90	0.87	1.0	0.87	1.0	0.91	0.91	0.88	0.94	0.91
Precision	0.86	0.87	0.86	0.86	0.89	0.85	0.88	0.87	1.0	0.87	1.0	0.85	0.88	0.84	0.91	0.85
F1 Score	0.88	0.88	0.89	0.88	0.90	0.87	0.89	0.87	1.0	0.87	1.0	0.88	0.90	0.86	0.93	0.88

Table 24. Comparison table of Blair for both training and testing records



	Logistic Train Hague	Logistic Test Hague	LDA Train Hague	LDA Test Hague	KNN Train Hague	KNN Test Hague	NB Train Hague	NB Test Hague	bgcl Train Hague	bgcl Test Hague	RF Train Hague	RF Test Hague	ADB Train Hague	ADB Test Hague	gb Train Hague	gb Test Hague
Accuracy	0.83	0.86	0.8	0.85	1.0	0.83	0.84	0.82	0.99	0.82	1.0	0.82	0.85	0.81	0.89	0.84
AUC Score	0.88	0.91	0.88	0.91	1.0	0.88	0.89	0.88	1.00	0.89	1.0	0.88	0.91	0.88	0.95	0.90
Recall	0.66	0.68	0.67	0.69	1.0	0.65	0.69	0.73	0.98	0.60	1.0	0.71	0.70	0.67	0.77	0.70
Precision	0.75	0.82	0.72	0.80	1.0	0.79	0.73	0.74	1.00	0.81	1.0	0.74	0.76	0.75	0.84	0.79
F1 Score	0.70	0.74	0.70	0.74	1.0	0.72	0.71	0.73	0.99	0.69	1.0	0.72	0.73	0.71	0.80	0.74

Table 25. Comparison table of Hague for both training and testing records

- Boosting Model performs better in all aspects.
- Logistic regression, LDA and Naïve Bayes' produces the consistent results.
- KNN Model has poor ROC curve compared to other models. Also, it is over fitting for Hague's records.
- Bagging and RF has a overfitting problem.

Based on the provided dataset, without doing any modifications in the dataset, for all the models, we have got the scores of 85% and above approximately, which is a good sign of our models are valid models.

KNN, Bagging and RF are over fitting the records. For this business objective, Gradient Boosting model appears to be the best optimized model compared to other valid models in all aspects. In Gradient Boosting, each predictor tries to improve on its predecessor by reducing the errors. But the fascinating idea behind Gradient Boosting is that instead of fitting a predictor on the data at each iteration, it actually fits a new predictor to *the residual errors made by the previous predictor.*

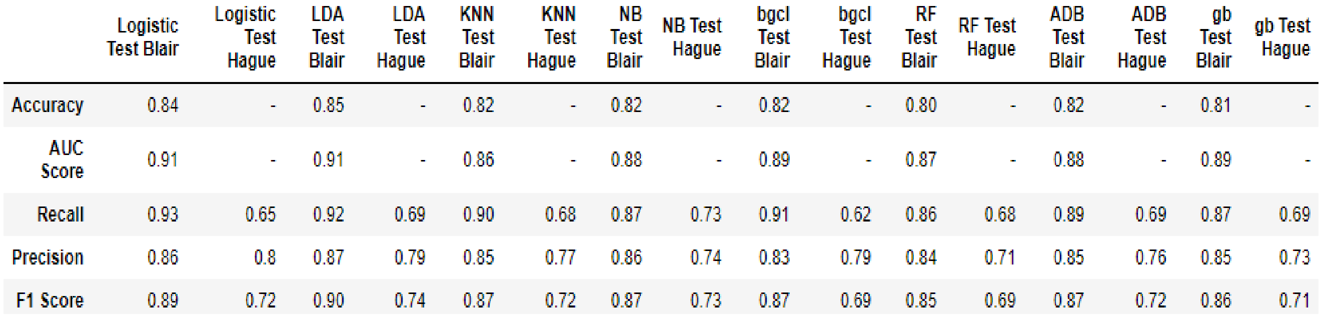
Binning

For the age Category, we group the column using 'binning' technique as shown below and rerun all models.

Age	Category	Percentage
20-30	Young	5%
30-40	Adults	18.79%
40-60	mid-age	39.22%
60-93	Seniors	36.98%

Table 26. Age grouping

With binning, actually the numbers are coming down slightly and test results are shown below.



	Logistic Test Blair	Logistic Test Hague	LDA Test Blair	LDA Test Hague	KNN Test Blair	KNN Test Hague	NB Test Blair	NB Test Hague	bgcl Test Blair	bgcl Test Hague	RF Test Blair	RF Test Hague	ADB Test Blair	ADB Test Hague	gb Test Blair	gb Test Hague
Accuracy	0.84	-	0.85	-	0.82	-	0.82	-	0.82	-	0.80	-	0.82	-	0.81	-
AUC Score	0.91	-	0.91	-	0.86	-	0.88	-	0.89	-	0.87	-	0.88	-	0.89	-
Recall	0.93	0.65	0.92	0.69	0.90	0.68	0.87	0.73	0.91	0.62	0.86	0.68	0.89	0.69	0.87	0.69
Precision	0.86	0.8	0.87	0.79	0.85	0.77	0.86	0.74	0.83	0.79	0.84	0.71	0.85	0.76	0.85	0.73
F1 Score	0.89	0.72	0.90	0.74	0.87	0.72	0.87	0.73	0.87	0.69	0.85	0.69	0.87	0.72	0.86	0.71

Table 27. Comparison table of Blair Vs Hague with binning for the testing records

SMOTE

SMOTE approach is to addressing imbalanced datasets is to oversample the minority class.

Before OverSampling, counts of label '1' in the training record: 739
Before OverSampling, counts of label '0' in the training record: 322

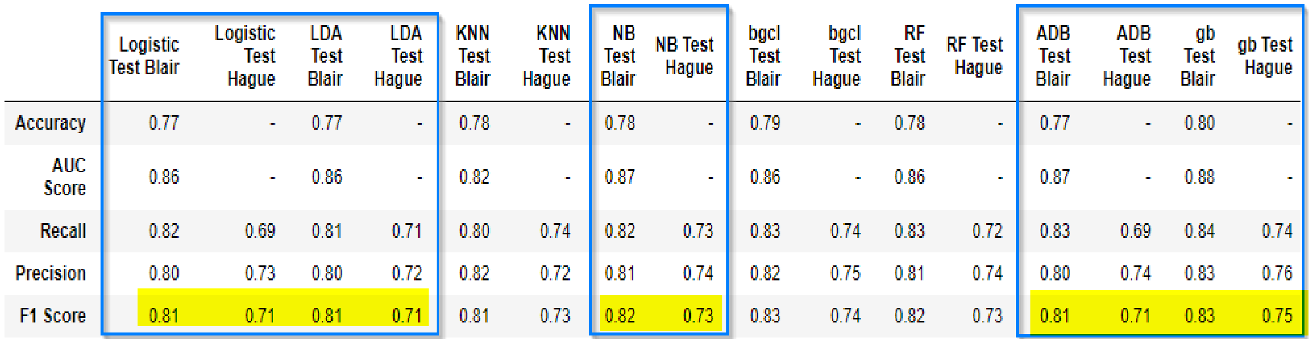
Impute the imbalance records using the SMOTE technique.

After OverSampling, counts of label '1': 739
After OverSampling, counts of label '0': 739

Now, the training records are increased from 1061 to 1478. After dropping the 312 duplicate records, the percentage of target value counts of Hague (target value 0) increased to 10% and rerun all the models.

	Train Labels original	Train Labels after SMOTE
Ratio	0 0.302632 1 0.697368	0 0.409314 1 0.590686

Table 28. Percentage of training labels before and after SMOTE



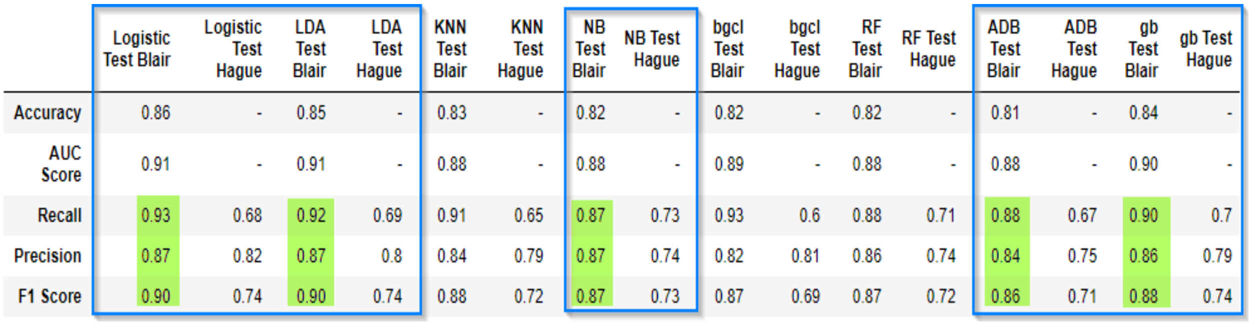
	Logistic Test Blair	Logistic Test Hague	LDA Test Blair	LDA Test Hague	KNN Test Blair	KNN Test Hague	NB Test Blair	NB Test Hague	bgcl Test Blair	bgcl Test Hague	RF Test Blair	RF Test Hague	ADB Test Blair	ADB Test Hague	gb Test Blair	gb Test Hague
Accuracy	0.77	-	0.77	-	0.78	-	0.78	-	0.79	-	0.78	-	0.77	-	0.80	-
AUC Score	0.86	-	0.86	-	0.82	-	0.87	-	0.86	-	0.86	-	0.87	-	0.88	-
Recall	0.82	0.69	0.81	0.71	0.80	0.74	0.82	0.73	0.83	0.74	0.83	0.72	0.83	0.69	0.84	0.74
Precision	0.80	0.73	0.80	0.72	0.82	0.72	0.81	0.74	0.82	0.75	0.81	0.74	0.80	0.74	0.83	0.76
F1 Score	0.81	0.71	0.81	0.71	0.81	0.73	0.82	0.73	0.83	0.74	0.82	0.73	0.81	0.71	0.83	0.75

Table 29. Comparison table of Blair Vs Hague with SMOTE and binning for the testing records

By looking at the results after running the models with SMOTE imputation of records, the model’s performance is coming down. So, let’s conclude our analysis with the original records (*without binning technique and without SMOTE*).

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.

Recommendations and Insights



	Logistic Test Blair	Logistic Test Hague	LDA Test Blair	LDA Test Hague	KNN Test Blair	KNN Test Hague	NB Test Blair	NB Test Hague	bgcl Test Blair	bgcl Test Hague	RF Test Blair	RF Test Hague	ADB Test Blair	ADB Test Hague	gb Test Blair	gb Test Hague
Accuracy	0.86	-	0.85	-	0.83	-	0.82	-	0.82	-	0.82	-	0.81	-	0.84	-
AUC Score	0.91	-	0.91	-	0.88	-	0.88	-	0.89	-	0.88	-	0.88	-	0.90	-
Recall	0.93	0.68	0.92	0.69	0.91	0.65	0.87	0.73	0.93	0.6	0.88	0.71	0.88	0.67	0.90	0.7
Precision	0.87	0.82	0.87	0.8	0.84	0.79	0.87	0.74	0.82	0.81	0.86	0.74	0.84	0.75	0.86	0.79
F1 Score	0.90	0.74	0.90	0.74	0.88	0.72	0.87	0.73	0.87	0.69	0.87	0.72	0.86	0.71	0.88	0.74

Table 30. Comparison table of Blair Vs Hague for the testing records

1. By looking at the results of the models such as Logistic, LDA, NB and Boosting, Blair would be winning the election. Labour party will cover the seats of approximately 20% higher than Conservative party.
2. During the campaign, one of the manifestos of a conservative party must give a guarantee to the public to increase the national & household economic conditions. This way, there is a chance for respondents to change their mind.
3. Based on the respondents' measures towards the European Integration, Labour party is consistent and average on the European Integration, where as Conservative party has got the surprised ratings of either low or high rating distribution.
4. Political knowledge for the Conservative party appears to be high.
5. In Labour party's manifesto, there is a point to be insisted to strengthen the European Integration relationship to win more seats.
6. There is no correlation between the Age and Voting.

Problem 2 – Inaugural corpora project

Executive Summary

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

Introduction

The purpose of the project is to extract the contents of inaugural speeches of the mentioned presidents of USA and to understand what they are conveying to the people.

Download the inaugural documents from nltk library. NLTK is a toolkit build for working with Natural Language Processing (NLP) in Python. It provides us various text processing libraries with a lot of test datasets.

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

The mentioned text documents are 1941-Roosevelt.txt, 1961-Kennedy.txt and 1973-Nixon.txt.

Using the nltk corpus package, let's import the inaugural corpus.

Using the corpus reader functions, we can extract the number of characters using the **raw()** function, words using **words()** function and sentences using the **sents()** function of the specific documents. To count the same, we can use the **len()** function.

Character, word and Sentence counts of US President's speeches

Documents	Character counts	Word counts	Sentence counts
1941-Roosevelt.txt	7571	1536	68
1961-Kennedy.txt	7618	1546	52
1973-Nixon.txt	9991	2028	69

Table 31. Character, word and sentence counts of the speeches

Total number of characters from all 3 speeches: 25180
Total number of words from all 3 speeches: 5110
Total number of sentences from all 3 speeches: 189

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.

- Stopwords are the English words which do not add much meaning to a sentence. They can safely be ignored without sacrificing the meaning of the sentence. Mostly they are words that are commonly used in the English language such as 'as, the, be, are' etc. NLTK (Natural Language Toolkit) in python has a list of stopwords stored in English language.
- There are additional punctuations such as **["--","`","``","'s",""""]** are found in the mentioned inaugural documents. Also, the common words like **[‘u’,’mr.’,mrs.’]** do not add meaning as well.
- We can add all these above words and punctuation to the existing stopwords list.
- Always constructs a lowercased words in the document using the **lower()** method.
- Using **‘nltk.tokenize’** library, let’s import **word_tokenize()** method, we are able to extract the tokens from string of characters by using **word_tokenize()** method. With this method, we will lemmatize each words present in the document.
- Download the ‘**wordnet** ‘corpus reader from nltk library for lemmatization. Python NLTK provides **WordNetLemmatizer()** that uses the WordNet Database to lookup lemmas of words. Lemmatization, unlike Stemming, reduces the inflected words properly ensuring that the root word belongs to the language. For example, *runs, running, ran* are all forms of the word *run*, therefore *run* is the lemma of all these words. Because lemmatization returns an actual word of the language, it is used where it is necessary to get valid words.

We define the above filters such as native stopwords, customized stopwords, punctuations, lemmatization in the user defined function called **content_fraction()** and pass the raw document text as the parameter to the function and finally return the word counts before and after the removal of stopwords and sample sentence after the removal of stopwords.

Word counts and Sample sentence after removal of stopwords

Documents	Word counts before removal of stopwords	Word counts after removal of stopwords	Sample Sentences after removal of stopwords
1941-Roosevelt.txt	1526	625	national day inauguration since 1789 people renew sense dedication united states washington day task people create weld together nation lincoln day task people preserve nation disruption...
1961-Kennedy.txt	1543	688	vice president johnson speaker chief justice president eisenhower vice president nixon president truman reverend clergy fellow citizens observe today victory party celebration freedom symbolize end well begin signify renewal well...
1973-Nixon.txt	2006	828	vice president speaker chief justice senator cook eisenhower fellow citizens great good country share together meet four years ago america bleak spirit depress prospect seemingly endless war abroad destructive...

Table 32. Word counts and Sample sentence after removal of stopwords

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)

NLTK in python has a function **FreqDist** which gives you the frequency of words within a text.

To find the top X most frequently used words in the document, use the **nltk.FreqDist.most_common()** function.

Define a user defined function called **top3words()** with the filters mentioned in the above question (please refer 2.2) and **nltk.FreqDist.most_common(3)** to the function and return the arguments to extract the top 3 words.

Top 3 words from the US president's speeches

Documents	Inaugural speeches	Top 3 words
1941-Roosevelt.txt	Top 3 words from 1941-Roosevelt's speech	`['nation', 'know', 'people']`
1961-Kennedy.txt	Top 3 words from 1961-Kennedy's speech	`['let', 'us', 'world']`
1973-Nixon.txt	Top 3 words from 1973-Nixon's speech	`['us', 'let', 'america']`

Table 33. Top 3 words from the US president's speeches

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

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. Word clouds are widely used for analyzing data from social network websites.

For generating word cloud in Python, modules needed are – matplotlib and wordcloud. Using the **wordcloud** package import **WordCloud**.

Pre-process the document individually with native stopwords, customized stopwords, punctuations and lemmatization. For detail information, please refer 2.2 answers.

Word cloud from 1941-Roosevelt's speech

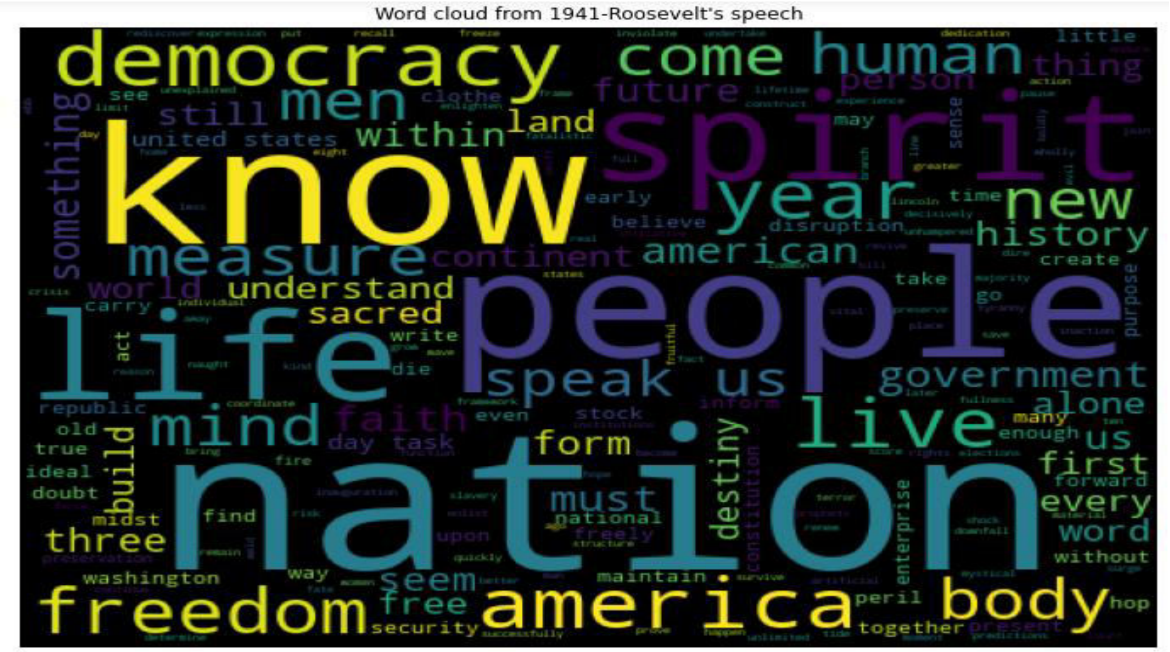


Figure 13.Word cloud from 1941-Roosevelt's Speech

Word cloud from 1961-Kennedy's speech

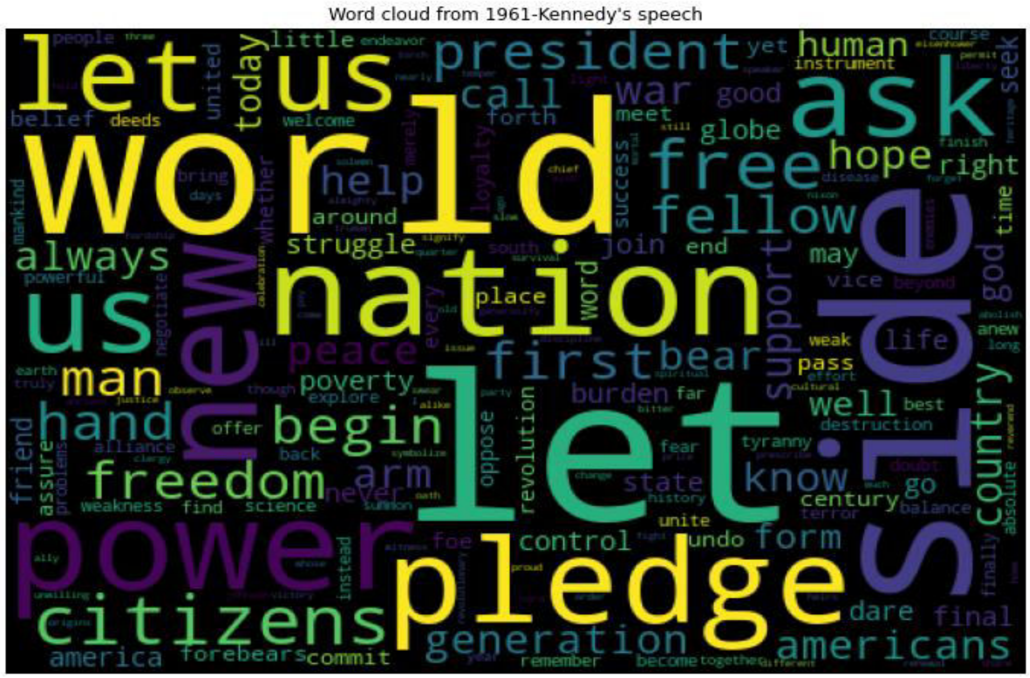


Figure 14. Word cloud from 1961-Kennedy's Speech

Word cloud from 1973-Nixon's speech

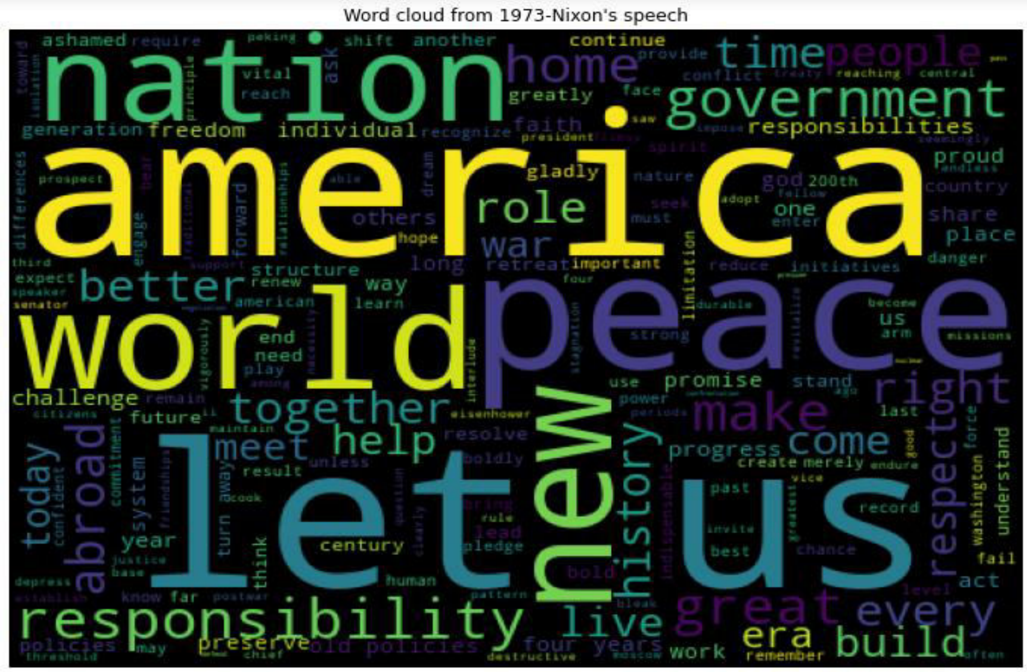


Figure 15. Word cloud from 1973-Nixon's Speech

THE END!