MACHINE LEARNING

PROJECT REPORT

DSBA

Table of Contents

Table of Figures

Problem Statement

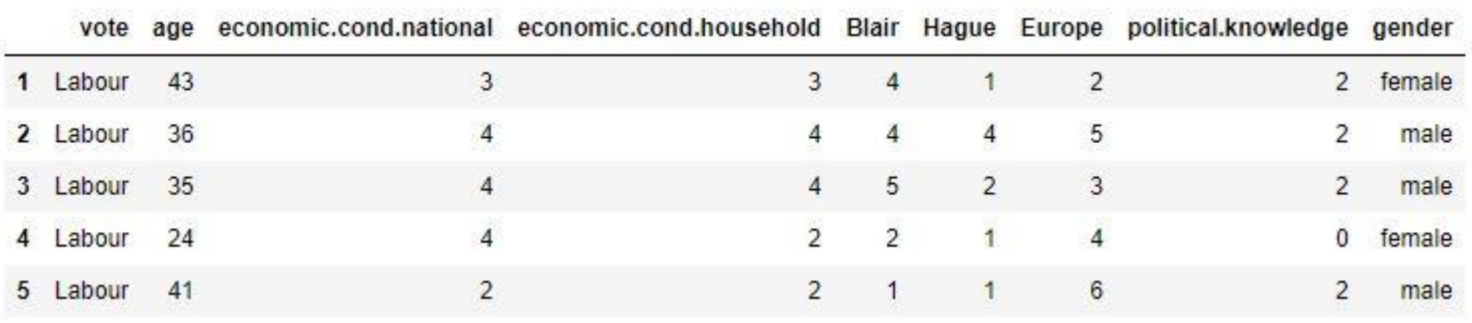
You are hired by one of the leading news channels CNBE who wants to analyze recent elections. This survey was conducted on 1525 voters with 9 variables. You have to build a model, to predict which party a voter will vote for on the basis of the given information, to create an exit poll that will help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

Data Dictionary

1. **vote:** Party choice: Conservative or Labour
2. **age:** in years
3. **economic.cond.national:** Assessment of current national economic conditions, 1 to 5.
4. **economic.cond.household:** Assessment of current household economic conditions, 1 to 5.
5. **Blair:** Assessment of the Labour leader, 1 to 5.
6. **Hague:** Assessment of the Conservative leader, 1 to 5.
7. **Europe:** an 11-point scale that measures respondents' attitudes toward European integration. High scores represent 'Eurosceptic' sentiment.
8. **political.knowledge:** Knowledge of parties' positions on European integration, 0 to 3.
9. **gender:** female or male.

1.1 Read the dataset. Do the descriptive statistics and do the null value condition check. Write an inference on it. (6 Marks)

<u>First 5 rows and last 5 rows of the dataset</u>



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

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

We will now conduct the checks such as info, descriptive statistics, skewness and checking for null and duplicate values

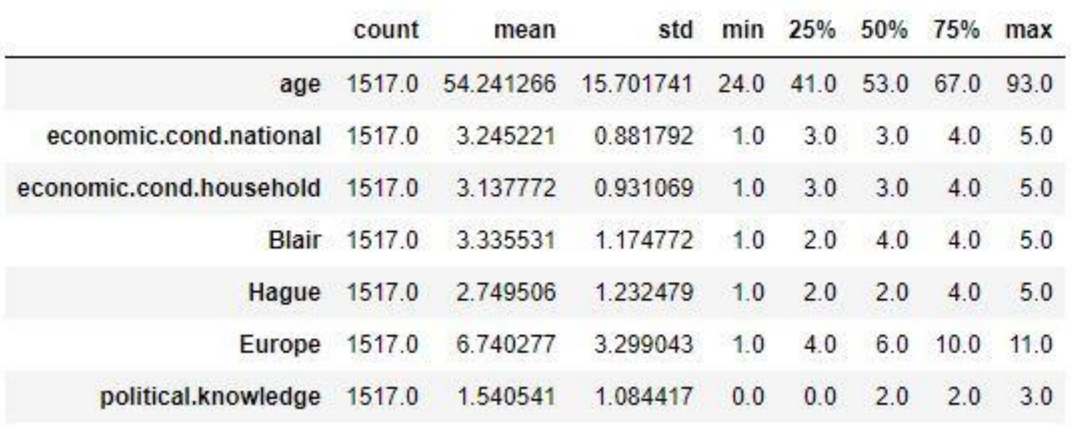
Basic info

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1525 entries, 1 to 1525
Data columns (total 9 columns):
 #   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
dtypes: int64(7), object(2)
memory usage: 119.1+ KB
```

Null Values in the data

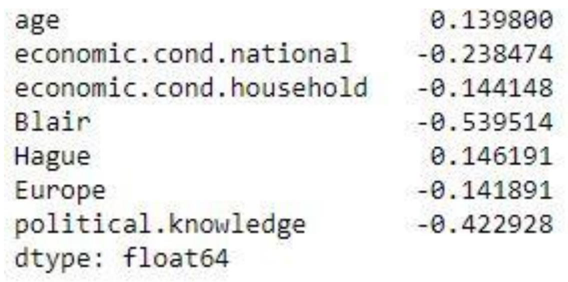
```
vote                       0
age                        0
economic.cond.national     0
economic.cond.household    0
Blair                      0
Hague                      0
Europe                     0
political.knowledge        0
gender                     0
dtype: int64
```

Total Number of Duplicate records = 8
So it has to be deleted so that the performance runs better



	count	mean	std	min	25%	50%	75%	max
age	1517.0	54.241266	15.701741	24.0	41.0	53.0	67.0	93.0
economic.cond.national	1517.0	3.245221	0.881792	1.0	3.0	3.0	4.0	5.0
economic.cond.household	1517.0	3.137772	0.931069	1.0	3.0	3.0	4.0	5.0
Blair	1517.0	3.335531	1.174772	1.0	2.0	4.0	4.0	5.0
Hague	1517.0	2.749506	1.232479	1.0	2.0	2.0	4.0	5.0
Europe	1517.0	6.740277	3.299043	1.0	4.0	6.0	10.0	11.0
political.knowledge	1517.0	1.540541	1.084417	0.0	0.0	2.0	2.0	3.0

Here we can see that mean and median for most of the columns are almost equal so we can say that the data is normally distributed but it slightly differs for age and political knowledge column



```
age                         0.139800
economic.cond.national     -0.238474
economic.cond.household    -0.144148
Blair                      -0.539514
Hague                       0.146191
Europe                     -0.141891
political.knowledge        -0.422928
dtype: float64
```

Here we can see that the data is slightly skewed

From the basic checks above, we can see the following

- There are a total of 1517 records in the dataset with 9 columns / features
- We have 7 quantitative features and 2 object features
- There were 8 null and duplicate values in the dataset which has been treated and removed
- We also see that vote and gender column are binary in nature which we have to convert into integer type for better workings of model as we know that all the models like KNN, Regression model or classification models runs better on quantitative data

1.2 Perform Univariate and Bivariate Analysis. Do exploratory data analysis. Check for Outliers. (8 Marks)

Univariate analysis

- Numerical variables - we will plot histogram and boxplots for each of the four numerical variables to understand distribution and check for outliers

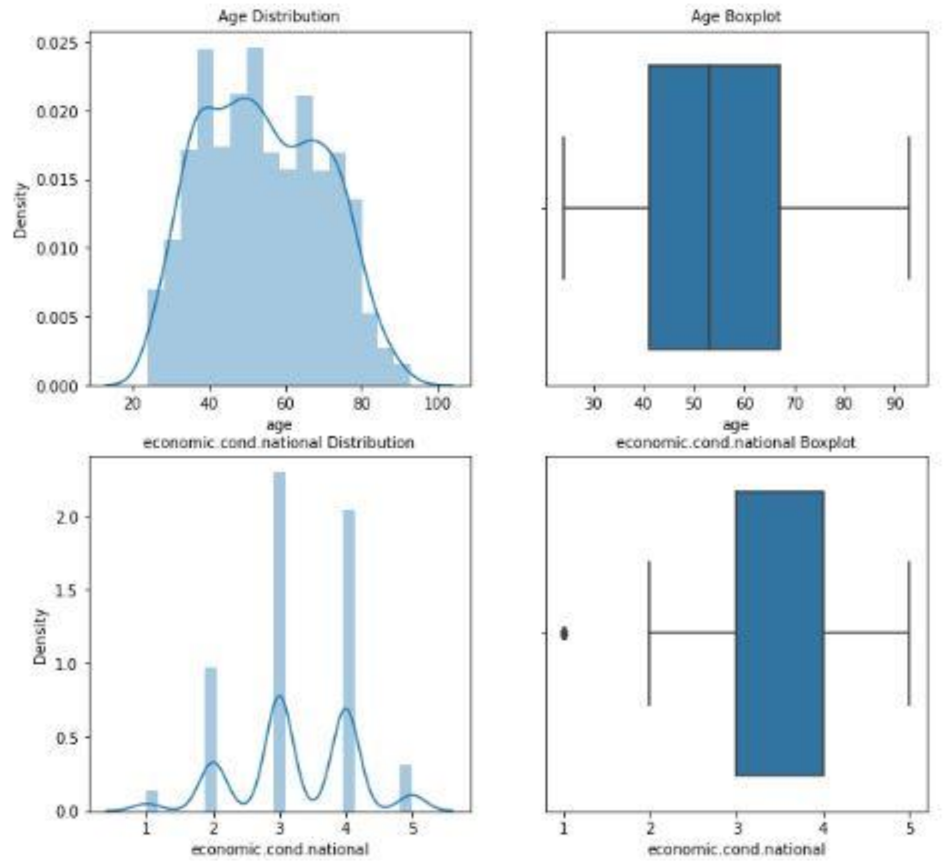


Figure 1 - Age distribution & economic condition national

We can see in this histogram and boxplot that the age of maximum voters are between 35 to 70 and in economic condition maximum voters have given 3 or 4 number in which we can see only one outlier which has to be treated otherwise the data is very good for evaluation of models and it is not a bad data at all.

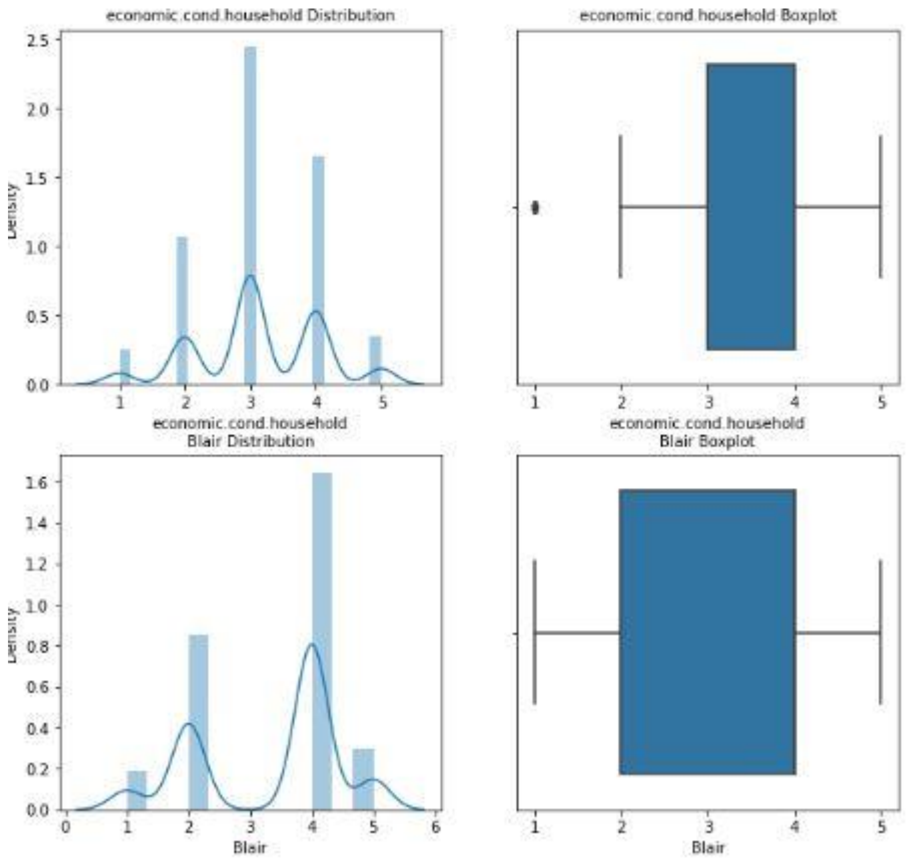


Figure 2 – Economic Condition household and Blair Distribution

Similarly for economic house hold condition we can see maximum numbers are between 3 to 4 which means overall economic condition of the voters are in good state and here also there is only one outlier which needs to be treated now in blair we can see huge number of voters have given 4 number to the leader of labour party which means maximum people are happy with him.

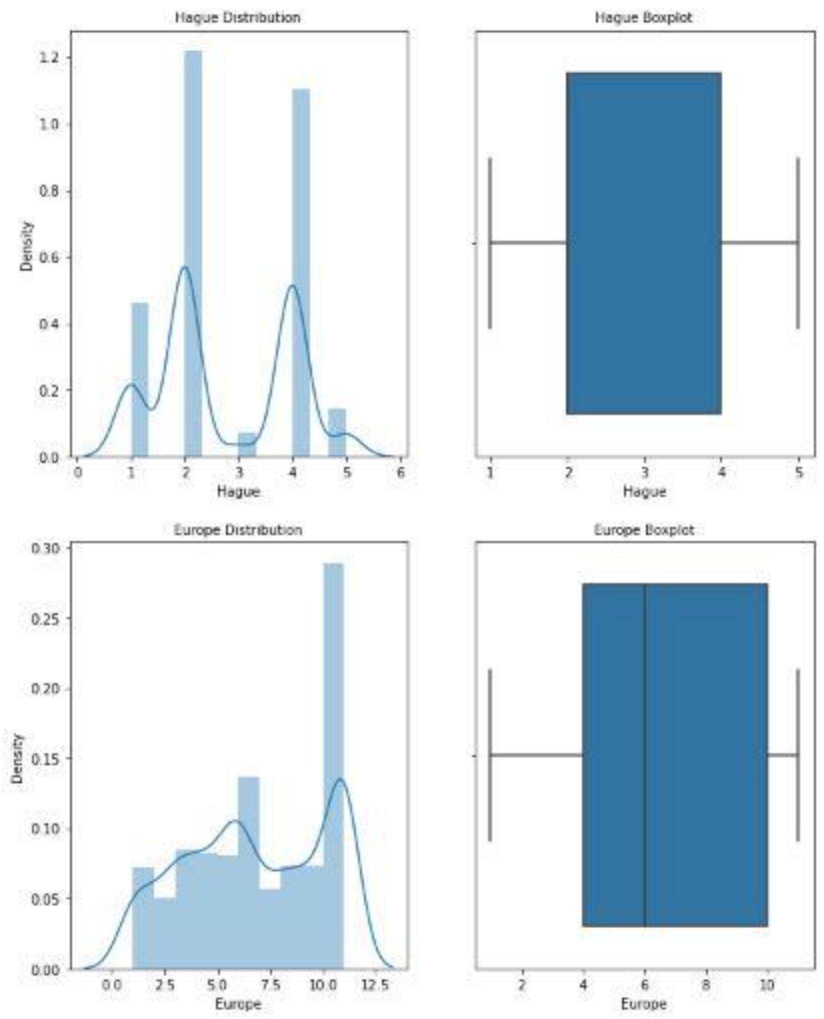


Figure 3 – Hague & Europe

Here in Hague we can see more number of voters have given only 2 rating to conservative party leader which means people are not that happy with his performance therefore with numbers we can also see there are more number of voters for labour party in comparison with conservative party. Europe this is related to the sentiments of the people which is majorly between 5 to 10. This is also a good distribution for model evaluation.

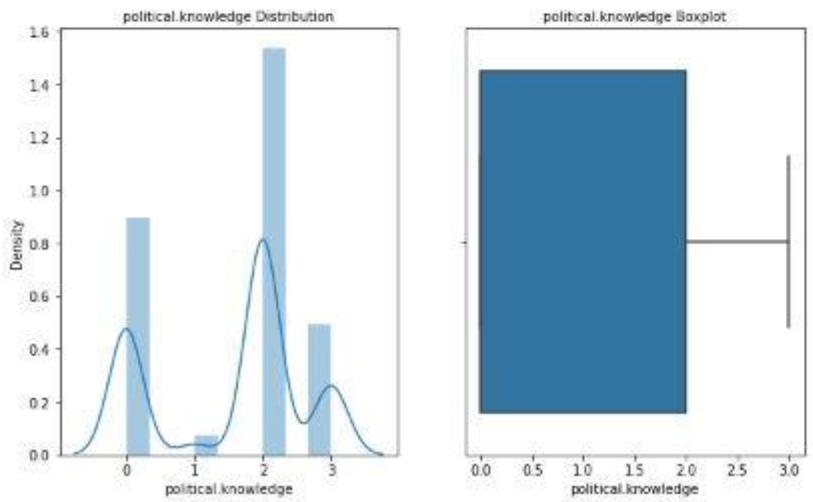
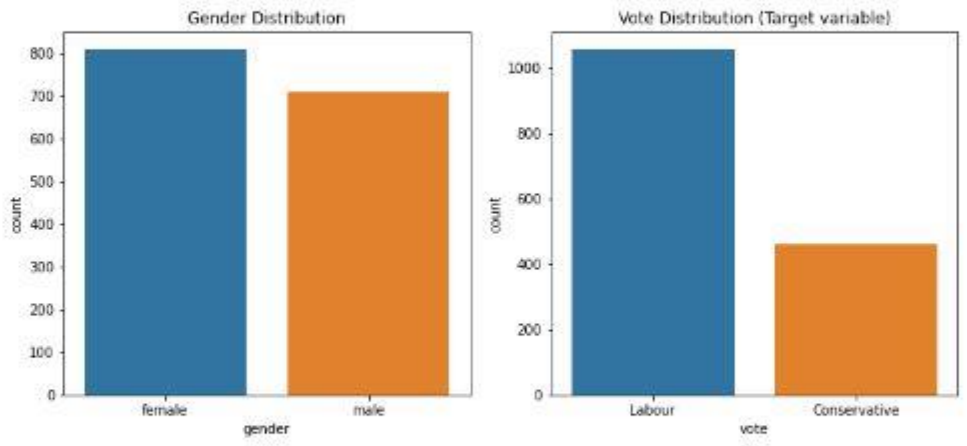


Figure 4 – Political Knowledge

Here we can see in political knowledge majority of people are between 0 to 2 which means majority of voters are not much connected to politics as their sentiments

Distribution for Categorical Variables



There are almost Equivalent number of voters between male and female which is around 700 (47%) for male and 800 (53%) for female and if we see vote distribution there are huge difference for both the parties which is 1057 around 70% for labour party and 460 around 30% for conservative party. From here we can interpret that labour party are getting more votes and are more favourate for the voters

However, when we build our machine learning models, we will see if model performance is able to predict the choice for voters for the future or not.

Bivariate analysis

- Numerical v numerical variables - we will plot a pairplot for the 7 numerical features and also a correlation heatmap to understand association between the numerical variables

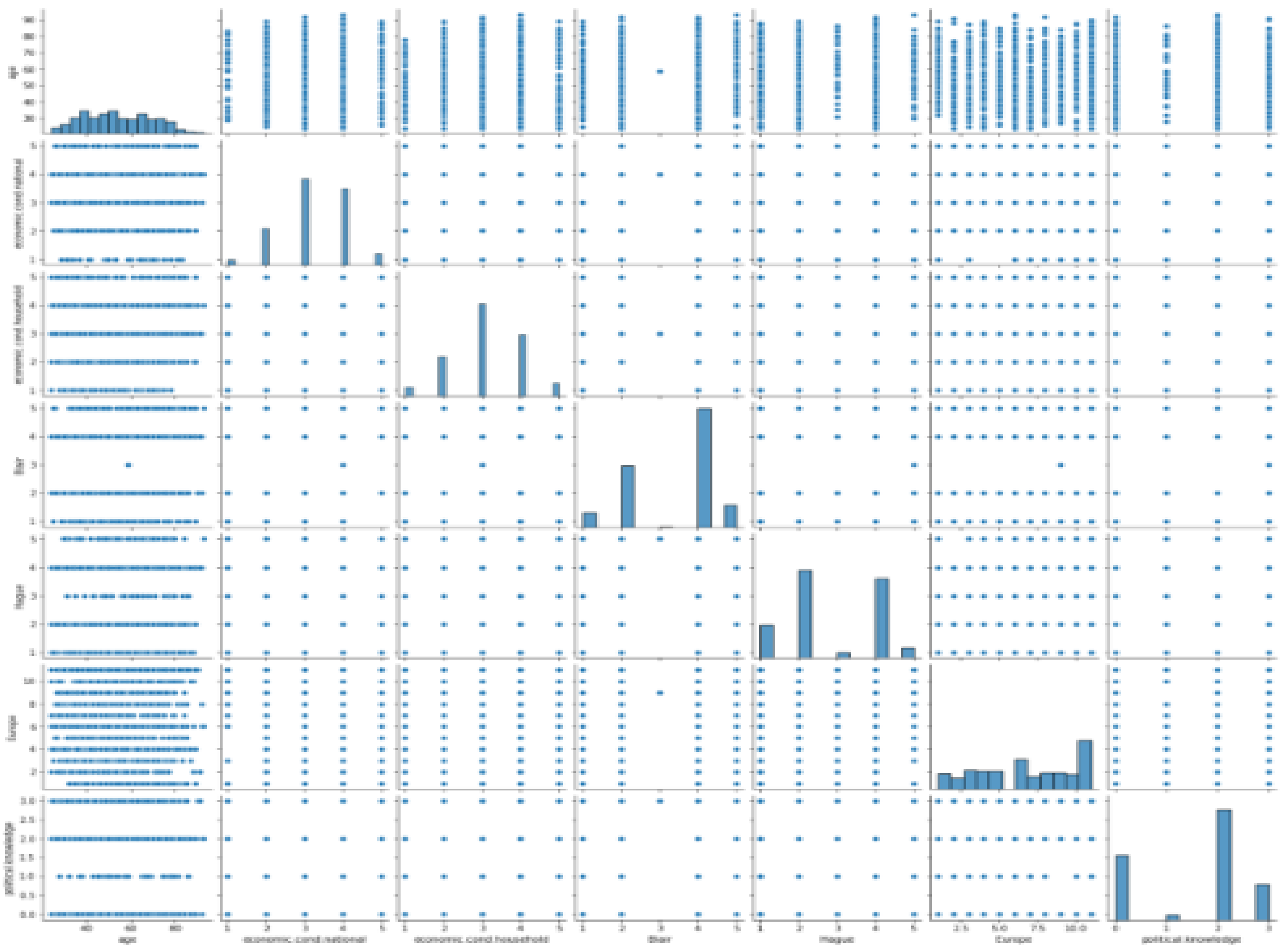
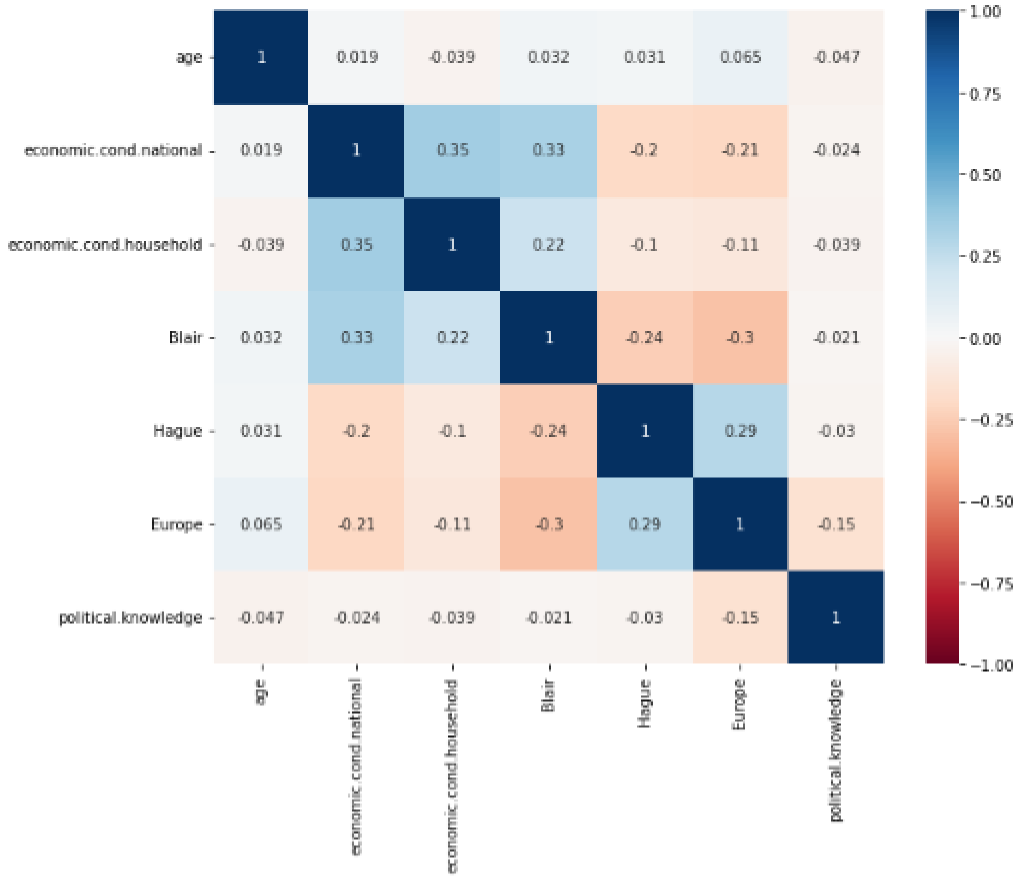


Figure 5 - Pairplot numerical variables



	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge
age	1	0.019	-0.039	0.032	0.031	0.065	-0.047
economic.cond.national	0.019	1	0.35	0.33	-0.2	-0.21	-0.024
economic.cond.household	-0.039	0.35	1	0.22	-0.1	-0.11	-0.039
Blair	0.032	0.33	0.22	1	-0.24	-0.3	-0.021
Hague	0.031	-0.2	-0.1	-0.24	1	0.29	-0.03
Europe	0.065	-0.21	-0.11	-0.3	0.29	1	-0.15
political.knowledge	-0.047	-0.024	-0.039	-0.021	-0.03	-0.15	1

Figure 6 - Heatmap all numerical variables

- In this heatmap and pairplot as we can clearly see that there is very less relation between these columns only economic conditions of both national and household are a bit related to each other but that is also not a really strong relation
- There are even parameters where the relation is also less than 10% which means almost no relation between the variables

- Categorical v categorical variables - we will plot countplots with hue for this

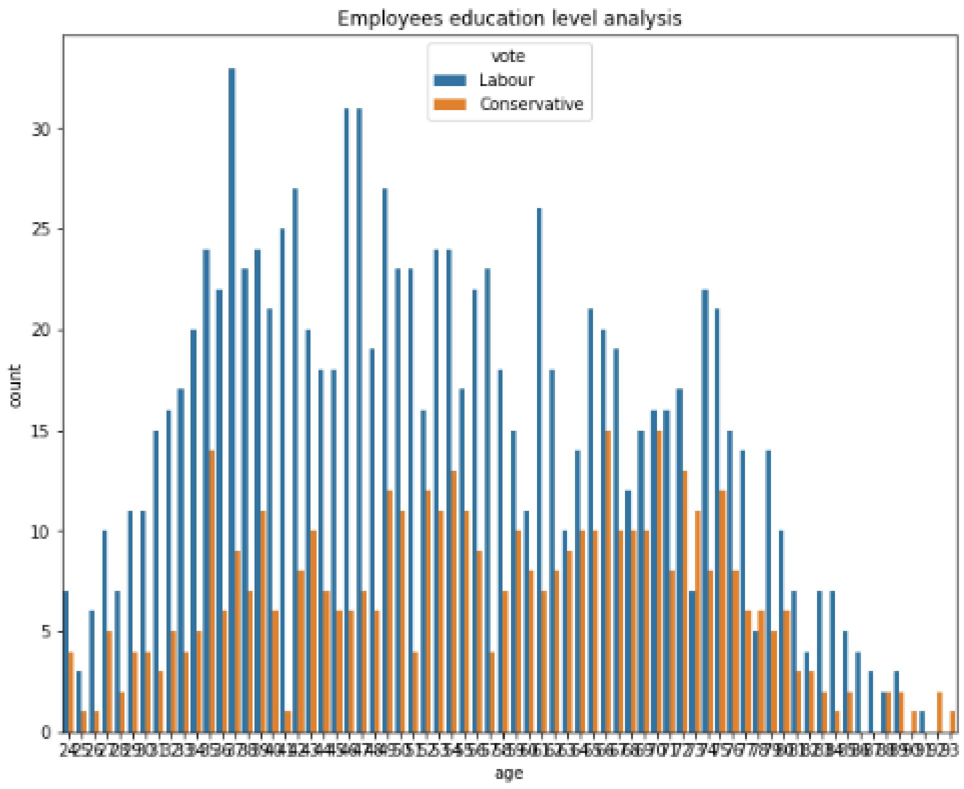


Figure 7 – Age Vs Vote

As We can see in this count plot all the age of the people support labour party majorly in comparison to conservative party.

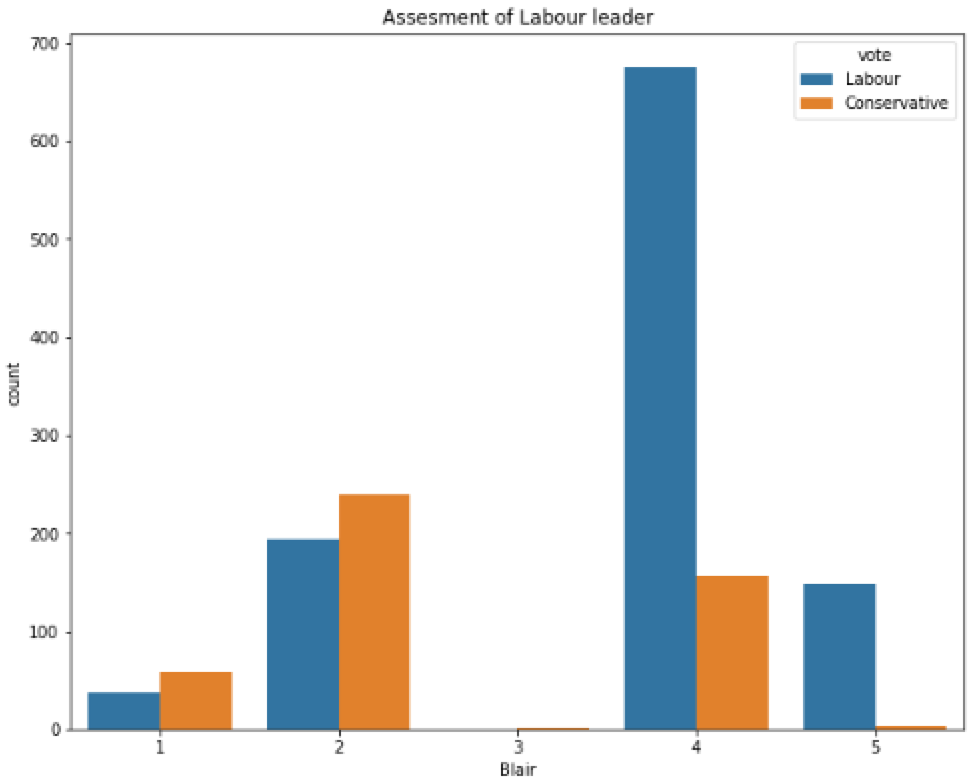


Figure 8 – Blair Vs Vote

Provided in the data Blair is the rating given to the leader of labour party by the voters as we can clearly see in blair the low ratings that is 1 & 2 is given majority to leader of conservative party and for the higher ratings that is 4 & 5 labour party has received huge number of ratings. Very few 5 ratings are been given to the leader of conservative party

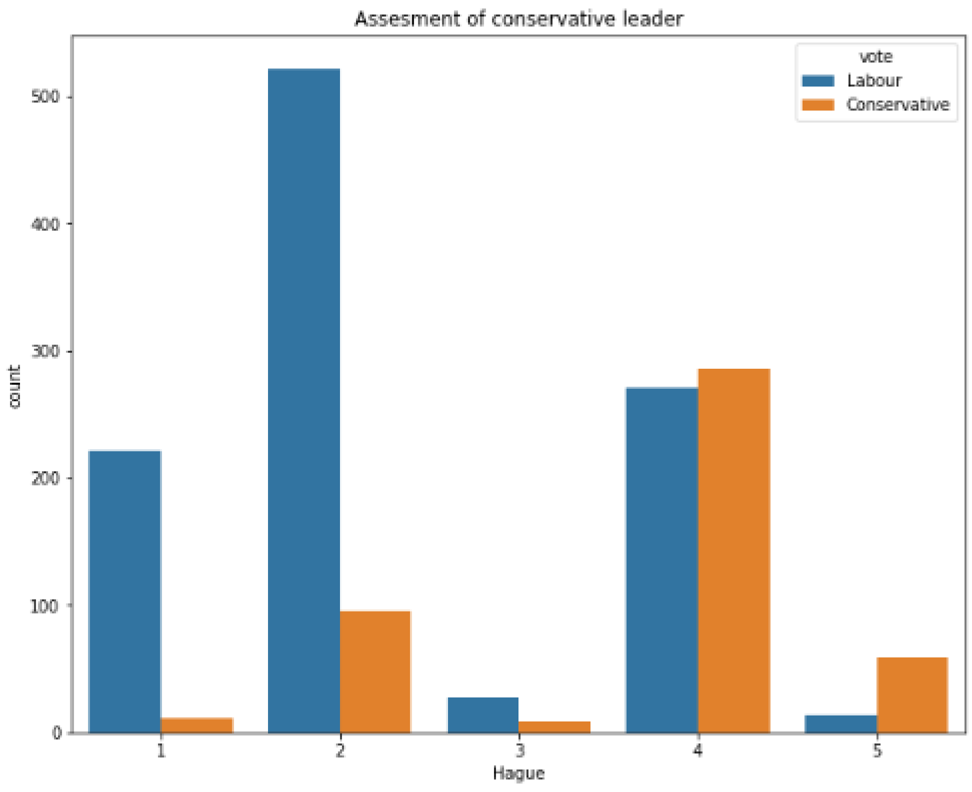


Figure 9 – Hague Vs Vote

In   Hague is the ratings given to Leader of conservative party by the voters in this low ratings are majorly provided to the leader of labour party that means people of this criteria don't majorly like the leader of labour party but for the higher ratings like 4 & 5 there is not much of a difference between the two leaders as we can see in the case of blair

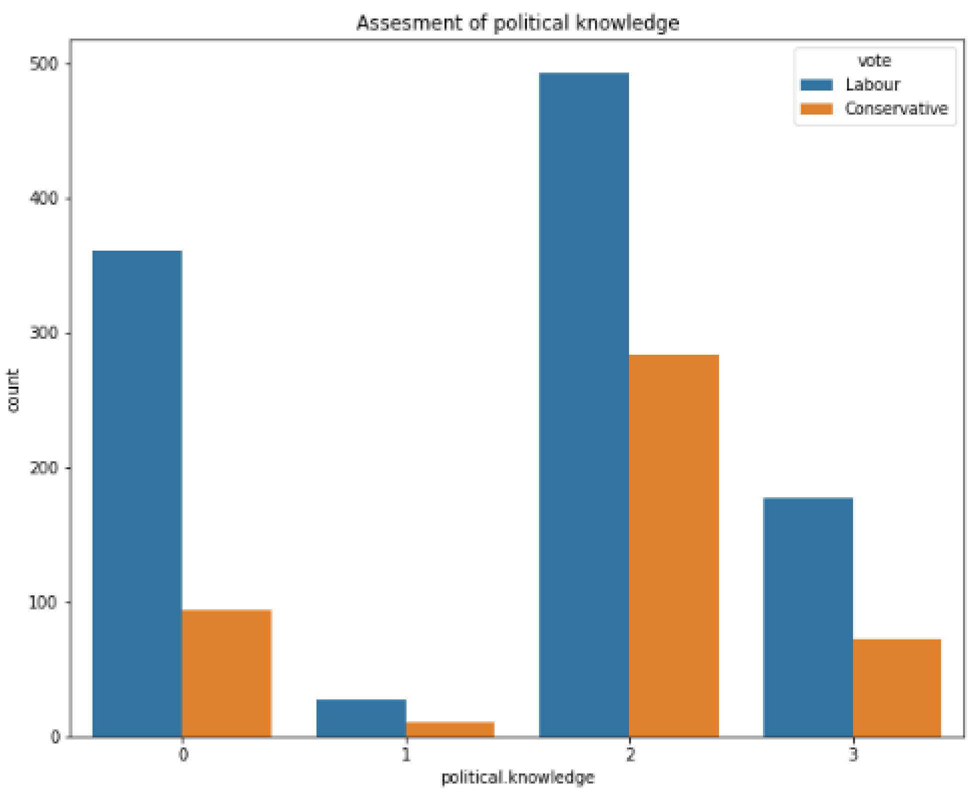


Figure 10 – Political knowledge Vs Vote

As we can interpret from this countplot that voters having political knowledge or not but more number of voters are voting to labour party which means there are more supporters for the labour party and people trust more for the leader of labour party.

EDA Summary

- With EDA we can clearly identify from this data that the labour party has a brighter future and has more number of supporters
- Despite of the age of voters, their political knowledge, their economic condition, gender every area have a marginal difference between the voters of both the parties and every where we can see a clear winner for the labour party

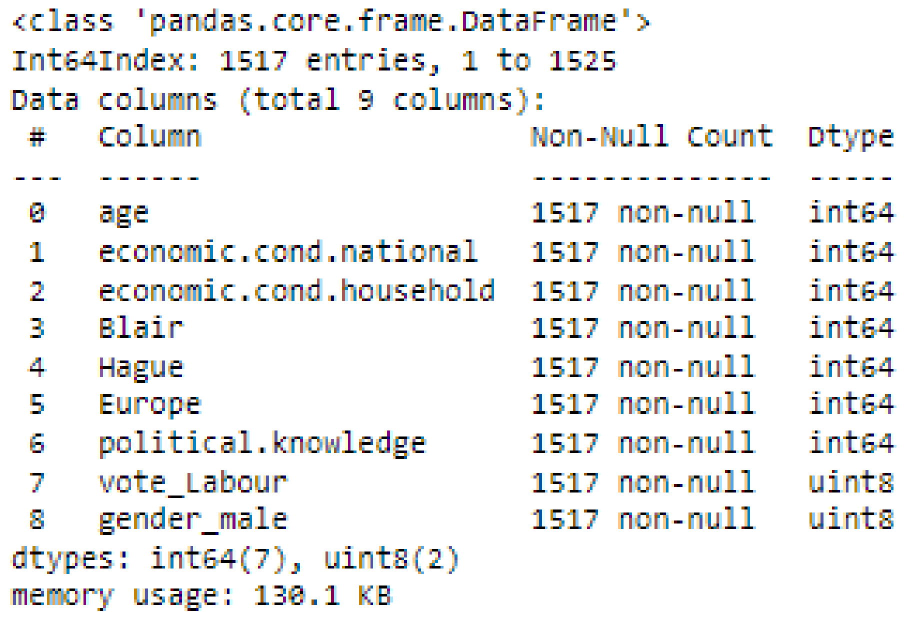
We will further build of model on this data to analyse the outcomes from the model that we are going to build and see what the model predicts for the future voters according to this split of data

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

All machine learning models only take numerical features as inputs, and hence it is required to convert all categorical / object features into numerical features either by using label encoding (typically used for binary value features or ordinal feature) or dummy / one hot encoding (used for nominal features)

We have 2 categorical features here all of them having binary values (0 or 1), we will simply convert them to numerical features. For the target variable we will assign 0 to Conservative Party and 1 to Labour Party and for gender we will assign 1 to Male and 0 to Female purely for a model building.

exercise<u>Conversion to numerical features</u>



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1517 entries, 1 to 1525
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   age                      1517 non-null   int64
 1   economic.cond.national   1517 non-null   int64
 2   economic.cond.household  1517 non-null   int64
 3   Blair                    1517 non-null   int64
 4   Hague                    1517 non-null   int64
 5   Europe                   1517 non-null   int64
 6   political.knowledge      1517 non-null   int64
 7   vote_Labour              1517 non-null   uint8
 8   gender_male              1517 non-null   uint8
dtypes: int64(7), uint8(2)
memory usage: 130.1 KB
```

We now see that all columns are of the numerical type and can be used for the various machine learning models

<u>Train - Test data split</u>

For all machine learning models we would split the data into train and test data typically as 70:30, 75:25 or 80:20 split with the model being built on training data and tested for prediction accuracy on the test data.

We will first split our dataset into X (dependent variables) and y (target variable i.e. Vote) and then use the 70:30 split for train and test data

<u>Scaling</u>

Scaling - Generally, good performance for distance-based models (such as clustering and KNN) usually requires pre-processing of data to make all variables similarly scaled and centered.

It is always advisable to bring all the features to the same scale for applying distance-based algorithms like KNN

We will use Z-score scaling (Standard scaler) for our data.
Z-score is a variation of scaling that represents the number of standard deviations away from the mean.
You would use z-score to ensure your feature distributions have mean = 0 and std = 1. It’s useful when there are a few outliers, but not so extreme as is the case for our dataset

The formula for calculating the z-score of a point, x, is as follows:
$(x-\mu)/\sigma$
where μ is the mean and σ is the standard deviation.

	count	mean	std	min	25%	50%	75%	max
age	1061.0	1.887693e-16	1.000472	-1.940665	-0.781546	-0.073196	0.828341	2.502624
economic.cond.national	1061.0	-1.715038e-16	1.000472	-2.622026	-0.289137	-0.289137	0.877307	2.043751
economic.cond.household	1061.0	4.645985e-17	1.000472	-2.295434	-0.163744	-0.163744	0.902100	1.967945
Blair	1061.0	6.550420e-17	1.000472	-2.018037	-1.161925	0.550300	0.550300	1.406413
Hague	1061.0	-2.878627e-16	1.000472	-1.404459	-0.593283	-0.593283	1.029070	1.840246
Europe	1061.0	1.611445e-17	1.000472	-1.736401	-0.815854	-0.202156	1.025240	1.332089
political.knowledge	1061.0	1.017094e-16	1.000472	-1.407526	-1.407526	0.452231	0.452231	1.382110
gender_male	1061.0	-3.201963e-17	1.000472	-0.936950	-0.936950	-0.936950	1.067292	1.067292

As we can see from the above, all features now have a mean very close to 0 and SD close to 1 which is the case in z-scorescaling

1.4 Apply Logistic Regression.

Logistic Regression is defined as a statistical approach, for calculating the probability outputs for the target labels. In its basic form it is used to classify binary data. Logistic regression is very much similar to linear regression where the explanatory variables(X) are combined with weights to predict a target variable of binary class(y). The main difference between linear regression and logistic regression is of the type of the target variable.

The equation of the Logistic Regression by which we predict the corresponding probabilities and then go on predict a discrete target variable is $1/1+e^{-z}$

Note: z = $\beta_0 + \sum_{i=1}^{n}(\beta_i X_1)$

In linear regression we have a straight best fit line that captures the data well. In logistic regression we have something called a sigmoid function line

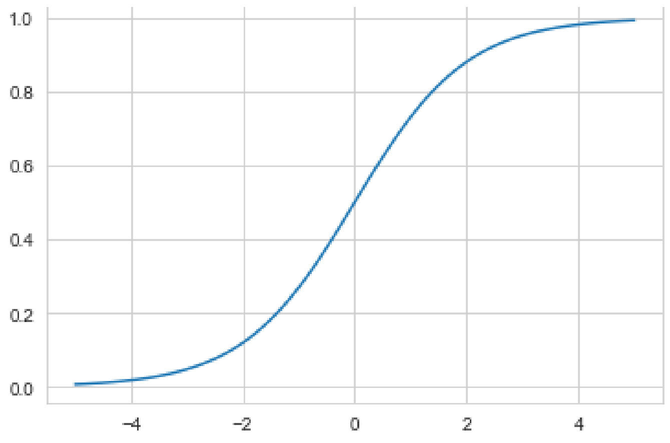


Figure 11 - Sigmoid line function

We will build a logistic regression model first with all independent variables included and then drop the variables based on VIF values (multi-collinearity) and insignificance of variables on the prediction model

1) **Base LR model with all features**

2) Logit Regression Results

Dep. Variable:	vote_Labour	**No. Observations:**	1517
Model:	Logit	**Df Residuals:**	1508
Method:	MLE	**Df Model:**	8
Date:	Sat, 15 Apr 2023	**Pseudo R-squ.:**	0.3811
Time:	02:19:49	**Log-Likelihood:**	-576.09
converged:	True	**LL-Null:**	-930.80
Covariance Type:	nonrobust	**LLR p-value:**	6.735e-148

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	3.0516	0.537	5.684	0.000	1.999	4.104
age	-0.0186	0.005	-3.896	0.000	-0.028	-0.009
economic_cond_national	0.4216	0.094	4.464	0.000	0.237	0.607
economic_cond_household	0.0704	0.086	0.817	0.414	-0.098	0.239
Blair	0.6223	0.066	9.403	0.000	0.493	0.752
Hague	-0.8404	0.066	-12.659	0.000	-0.971	-0.710
Europe	-0.2145	0.025	-8.544	0.000	-0.264	-0.165
political_knowledge	-0.4110	0.071	-5.755	0.000	-0.551	-0.271
gender_male	0.0983	0.151	0.651	0.515	-0.198	0.394

Figure 12 - Logistic regression base model

Interpretation of p-values (P > |t|)

- For each predictor variable there is a null hypothesis and alternate hypothesis.

- Null hypothesis : Predictor variable is not significant
- Alternate hypothesis : Predictor variable is significant

- (P > |t|) gives the p-value for each predictor variable to check the null hypothesis.
- If the level of significance is set to 5% (0.05), the p-values greater than 0.05 would indicate that the corresponding predictor variables are not significant.
- However, due to the presence of multicollinearity in our data, the p-values will also change.
- We need to ensure that there is no multicollinearity to interpret the p-values.

<u>How to check for Multicollinearity</u>

- There are different ways of detecting (or testing) multicollinearity. One such way is Variation Inflation Factor.
- **Variance Inflation factor**: Variance inflation factors measure the inflation in the variances of the regression coefficients estimates due to collinearities that exist among the predictors. It is a measure of how much the variance of the estimated regression coefficient βk is "inflated" by the existence of correlation among the predictor variables in the model.
- **General Rule of Thumb**:
    - If VIF is 1, then there is no correlation among the kth predictor and the remaining predictor variables, and hence, the variance of βk is not inflated at all.
    - If VIF exceeds 5, we say there is moderate VIF, and if it is 10 or exceeding 10, it shows signs of high multi-collinearity.
    - The purpose of the analysis should dictate which threshold to use.

<u>VIF values</u>

```
age  VIF =  1.01
economic_cond_national  VIF =  1.15
Blair  VIF =  1.22
Hague  VIF =  1.14
Europe  VIF =  1.21
political_knowledge  VIF =  1.03
```

As we can see in this data we don’t have any VIF value where the VIF value is more than 5

therefore we are not dropping any column in this data based on its VIF value

2) Revised model after VIF and p-value checks (dropping Gender)

Logit Regression Results

Dep. Variable:	vote_Labour	**No. Observations:**	1517
Model:	Logit	**Df Residuals:**	1509
Method:	MLE	**Df Model:**	7
Date:	Sat, 15 Apr 2023	**Pseudo R-squ.:**	0.3809
Time:	02:19:50	**Log-Likelihood:**	-576.30
converged:	True	**LL-Null:**	-930.80
Covariance Type:	nonrobust	**LLR p-value:**	7.957e-149

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	3.0846	0.535	5.766	0.000	2.036	4.133
age	-0.0187	0.005	-3.920	0.000	-0.028	-0.009
economic_cond_national	0.4235	0.094	4.486	0.000	0.238	0.609
economic_cond_household	0.0694	0.086	0.805	0.421	-0.100	0.238
Blair	0.6238	0.066	9.427	0.000	0.494	0.753
Hague	-0.8397	0.066	-12.656	0.000	-0.970	-0.710
Europe	-0.2146	0.025	-8.552	0.000	-0.264	-0.165
political_knowledge	-0.4057	0.071	-5.720	0.000	-0.545	-0.267

Figure 13 - LR model after dropping Gender column

As we can see above the gender column was having a p-value of more than 0.05 therefore we had to drop it to avoid multicollinearity and for better working of the model.

We will now drop Economic Condition household as well given p-value is greater than 0.05

3) Revised model after VIF and p-value checks after droping economic condition household as well

Logit Regression Results

Dep. Variable:	vote_Labour	**No. Observations:**	1517
Model:	Logit	**Df Residuals:**	1510
Method:	MLE	**Df Model:**	6
Date:	Sat, 15 Apr 2023	**Pseudo R-squ.:**	0.3805
Time:	02:19:50	**Log-Likelihood:**	-576.62
converged:	True	**LL-Null:**	-930.80
Covariance Type:	nonrobust	**LLR p-value:**	9.670e-150

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	3.2247	0.507	6.363	0.000	2.231	4.218
age	-0.0190	0.005	-4.001	0.000	-0.028	-0.010
economic_cond_national	0.4473	0.090	4.986	0.000	0.271	0.623
Blair	0.6295	0.066	9.570	0.000	0.501	0.758
Hague	-0.8406	0.066	-12.673	0.000	-0.971	-0.711
Europe	-0.2145	0.025	-8.550	0.000	-0.264	-0.165
political_knowledge	-0.4069	0.071	-5.740	0.000	-0.546	-0.268

Figure 14 - Best LR model after p-value and VIF checks

P-values for all remaining variables are less than 0.05, we will double check VIF values for the remaining variables to ensure there is no multicollinearity

```
age  VIF =  1.01
economic_cond_national  VIF =  1.15
Blair  VIF =  1.22
Hague  VIF =  1.14
Europe  VIF =  1.21
political_knowledge  VIF =  1.03
```

- All VIF values are less than 5 and p-values less than 0.05, this is our final logistic regression model.

- The final model has 6 independent variables - Age of voters, economic_cond_national, Blair, Hague,Europe and political_knowledge of voters

- Accuracy scores for both the base logistic regression model and final logistic regression model which we will now build on training data and predict both on training and testing data for evaluation.

Accuracy scores

```
Accuracy Score - Best Model
Training: 0.8284637134778511
Testing:  0.8552631578947368
```

We have a accuracy score of 82.85% for the training data and 85.53% for the testing data according to this matrics

Basis the accuracy scores, we can see no overfitting and underfitting of the data. The accuracy is the highest when it comes to the best model on test data. We will further look into model evaluation metrics in detail in Q1.7

4) Logistic regression model on dataset without outliers

We will also check for the logistic regression model performance with outliers treated to see if there is an impact of outliers on the model

Boxplot of dataset without outliers

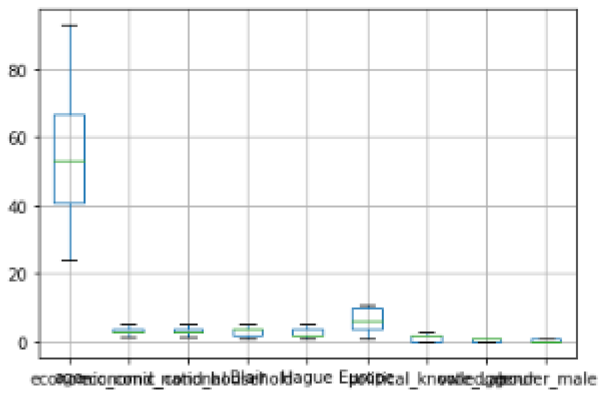


Figure 15 - Box plot for dataset without outliers

We see no outliers in a copy of the dataset we have made.

1.5 Apply KNN Model and Interpret the results.

KNN algorithm also known as K-Nearest Neighbours Algorithm is used to solve both problems of classification as well as regression. This working principle of algorithm is mainly based on feature similarity in both classification and regression problem. KNN classifier is different from other probabilistic classifiers where the model comprises a learning step of computing probabilities from a training sample and use them for future prediction of a test sample. In probability-based model once the model is trained the training sample could be thrown away and classification is done using the computed probabilities.

In KNN, the term `k' is a parameter which refers to the number of nearest neighbours. The classification procedure for a query point q works in two steps as:

- Find the K neighbours in the dataset which are closet to q based on the similarity measure.
- Use these K neighbours to determine the class of q using majority voting.

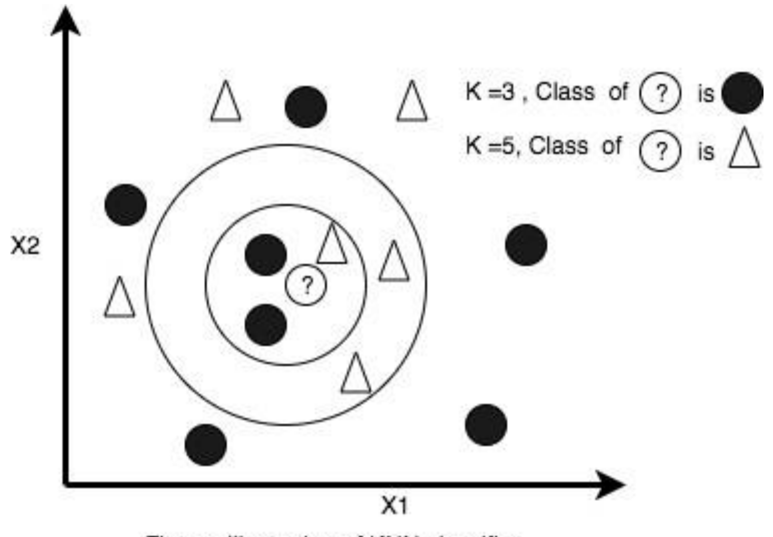


Figure: Illustration of KNN classifier

Figure 16 - Illustration of KNN classifier

Distance Measure- KNN classifier needs to compute similarities or distances of test query from each sample point in training dataset. Several methods are used to compute the distances and the choice completely depends of the types of features in the dataset. The popular distance measurements are as Euclidean distance, minkowski distance, chebyshev distance, manhattan/city block distance.

Given this being a distance-based algorithm, we will use the scaled data to build the model.

1) **KNN model (default K of 5) on scaled data** – the model is built on scaled data with the default parameters of the KNN model and K=5 i.e. 5 nearest neighbours

```
Accuracy (K=5) on train scaled data 0.8539114043355325
Accuracy (K=5) on test scaled data 0.8245614035087719
```

The accuracy scores of 0.85 on train data and 0.82 on test data shows no underfitting or overfitting

The default K is 5 in the above, however we will now check for the optimal number of K to maximise model performance

We will run the KNN with no of neighbours to be 1,3,5..19 and find the optimal number of neighbours from K=1,3,5,7... 19 using the Mis classification error

Misclassification error (MCE) = 1 - Test accuracy score. Calculated MCE for each model with neighbours = 1,3,5.. 19 and we will find the model with lowest MCE

We will plot the various MCE values to find the K value where MCE is the lowest.

MCE plot

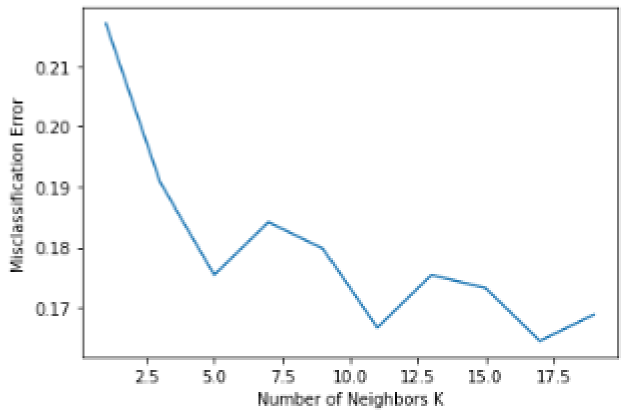


Figure 17 - MCE plot for optimal K in KNN

We see 12 as the optimal value for K and will now build a revised KNN model with 12 neighbors (K).

2) **KNN model with 12 as value of K on scaled data** – model built with all other parameters remaining asdefault values and K=12 i.e. 12 nearest neighbours

```
Accuracy (K=12) on train scaled data 0.8482563619227145
Accuracy (K=12) on test scaled data 0.8399122807017544
```

We see a small drop in train data accuracy and a small increase in test data accuracy when it comes to 12 neighbors v default value of 5 for neighbours.

Before we move for hyperparameter tuning, we will also check the KNN model performance for our dataset with outliers treated and see if there is a significant change in performance

3) **KNN model after hyperparameter tuning**

We will look at the following parameters to find the optimal combination of values for our KNN model

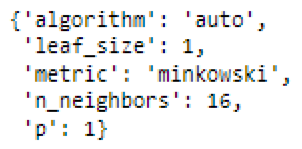
- n_neighbors: Decide the best k based on the values we have computed earlier. We will use a range of 3,5...19
- Algorithm - There are various algorithms such as auto, ball tree, kd tree and brute force algorithms that are used for KNN. We will keep all 4 options
- leaf_size - It is tuned when kd tree or ball tree is used. The leaf size controls the minimum number of points in a given node, and effectively adjusts the tradeoff between the cost of node traversal and the cost of a brute-force distance estimate. We have kept a range of 5 to 29 for this
- p - p is the value of p for Minkowski distance. If p is 1, it is Manhattan distance, if p is 2 it is

euclidean distance. We will keep p values of 1,2,3

- Metric - distance method to be used

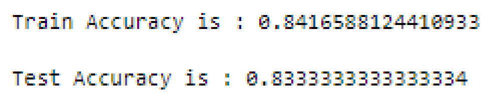
We will now perform the grid search with the above parameters and also include cross validation. Cross-validation is a resampling method that uses different portions of the data to test and train a model on different iterations. We will use default cv parameter of 5 in our grid search which will be a 5 fold cross validation. A CV value between 3 and 10 is considered a good option

<u>Best parameters after the gridsearch</u>



```
{'algorithm': 'auto',
 'leaf_size': 1,
 'metric': 'minkowski',
 'n_neighbors': 16,
 'p': 1}
```

<u>Model accuracy scores</u>



```
Train Accuracy is : 0.8416588124410933

Test Accuracy is : 0.8333333333333334
```

We see this model doing slightly better than the default KNN model (K=5) on test data and slightly worse on train data. However, when comparing this to the KNN model with 12 as value of K, the accuracy scoresare slightly lower on both training and test data. We will go into the detailed model evaluation - confusion matrix, classification report etc. in Q 1.7

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

Both Bagging and Boosting are ensemble techniques wherein they use multiple machine learning models on a dataset to make predictions. The way the data is used and method of voting of the variousmodels differ from technique to technique

Bagging

Bagging, also known as Bootstrap aggregating, is an ensemble learning technique that helps to improve the performance and accuracy of machine learning algorithms. It is used to deal with bias-variance trade-offs and reduces the variance of a prediction model. Bagging avoids overfitting of data and is used for both regression and classification models, specifically for decision tree algorithms.

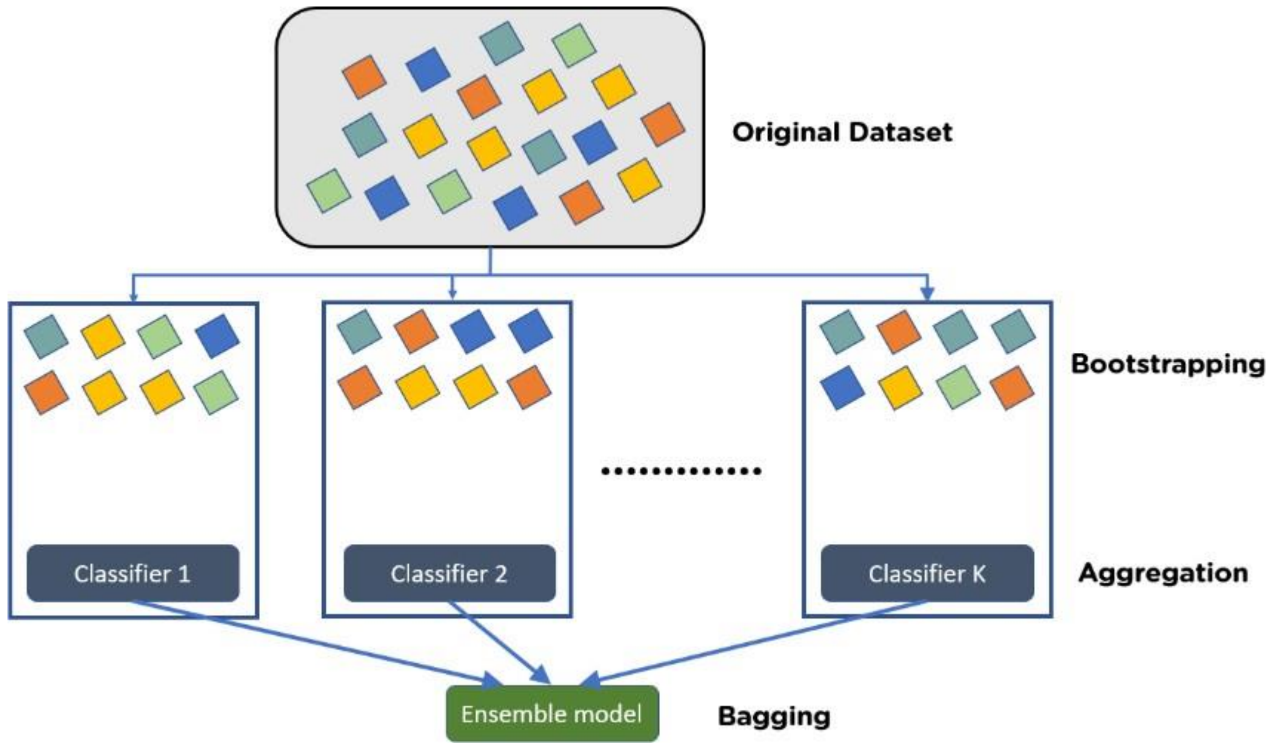


Figure 18 - Illustration bagging classifier

What Is Bootstrapping? Bootstrapping is the method of randomly creating samples of data out of a population with replacement to estimate a population parameter.

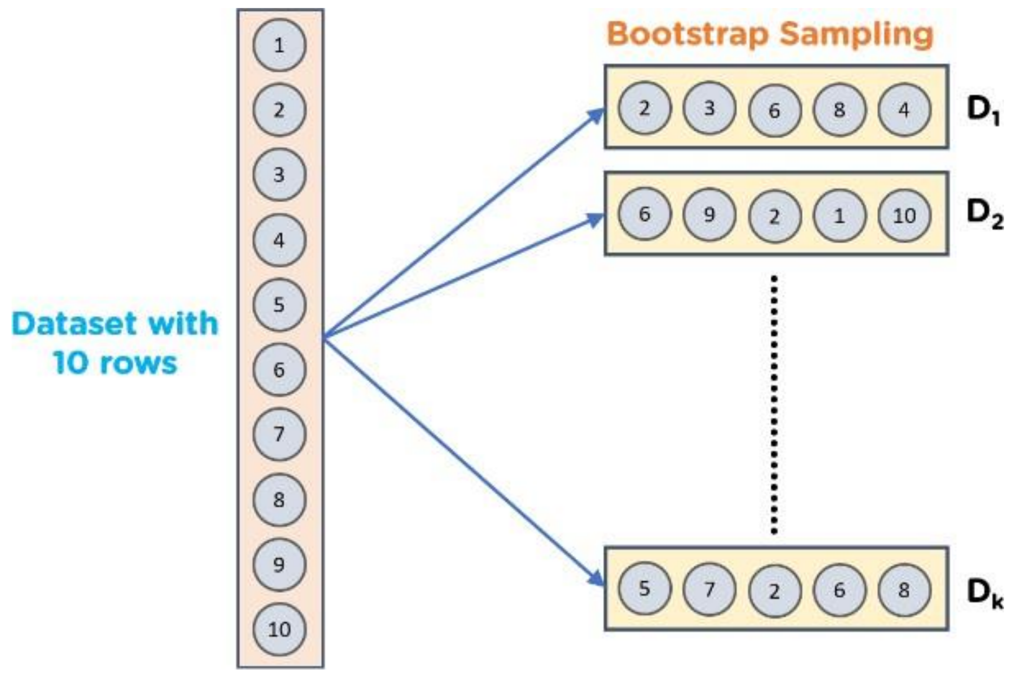


Figure 19 - Illustration Bootstrap sampling

Bagging – We will be using Random Forest algorithm as our bagging classifier

Random Forest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

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

RF uses the concept of bootstrapping. Bootstrapping is a method of sample selection. The formal definition describes it as “random sampling with replacement”. The techniques applied while creating random forests is to create several subsets of data from a single training sample by choosing samples randomly with replacement. Now, each collection of subset data is used to train the random forest.

We will first build a random forest model without any hyperparameter tuning and see model performance. Also we will use the unscaled data as used in logistic regression as random forest model does not require any feature scaling

1) **Base Random Forest model (Bagging model) without hyper parameter tuning – default parameters**

```
Accuracy of base RF model on train data 1.0
Accuracy of base RF model on test data 0.8289473684210527
```

We see a massive case of overfitting here as train data accuracy is 100% and 82.9% on test data. We will now do the hyperparameter tuning to refine our RF model.

2) Random Forest model after hyperparameter tuning

We will use parameters as follows to do a grid search and obtain the optimal parameters for the random forest.

- min_samples_split: The minimum number of samples required to split an internal node. This can be around 1% of the total number of records i.e. 4 for this data set. We will use a range of 3 to 10 in the grid search
- min_samples_leaf: The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This can be taken as 1/3 of min_samples_split so we will use a range of 1 to 5
- max_depth: The tree keeps developing to the maximum possible level in terms of depth. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples. We will use depths as 4 to 10
- n_estimators: Number of decision trees to be built. Given 444 records, we will keep this as 51, 71, 101 and 151
- max_features default=None The number of features to consider when looking for the best possible split. The algorithm shall selectmax_featuresat random at each split before finding the best split among them. There are a total of 9 columns, we will keep this between 3,4,5,6 and 7
- Criterion: Here we can define the measure of impurity that works best. We will keep 'gini' and 'entropy' as the two options

<u>Best parameters after grid search</u>

```
{'criterion': 'entropy',
 'max_depth': 5,
 'max_features': 5,
 'min_samples_leaf': 3,
 'min_samples_split': 8,
 'n_estimators': 71}
```

<u>Model accuracy scores</u>

```
Accuracy of best RF model on train data 0.8689915174363808
Accuracy of best RF model on test data 0.831140350877193
```

- We see a significant improvement from the base RF model accuracy scores - the best training data accuracy amongst all models built so far. The test data accuracy is also good, and better than most of the other models built except KNN base model with K=12
- There does not seem to be overfitting as difference in test and train data accuracy is less than 10%

Feature importance in the tuned Random Forest model

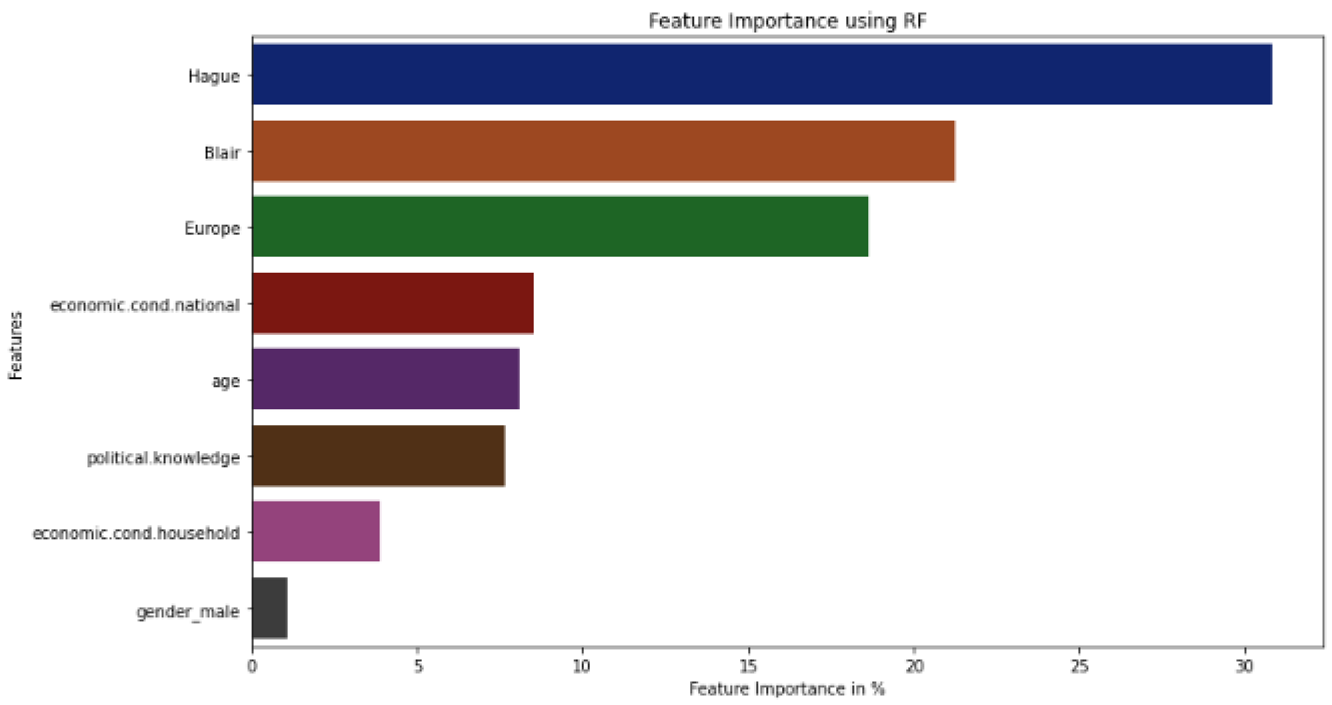


Figure 20 - Feature importance in tuned Random Forest model

In this model as well we can see the gender column and economic condition household column are the least important as we can see in the Logistic Regression model as well. In this we can see Hague is the most important column followed by blair, Europe, economic condition national and others.

For the detailed performance metrics (confusion matrix, ROC curve etc.) of the models built using Random Forest, we will delve further in Q1.7

Boosting

Boosting is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.

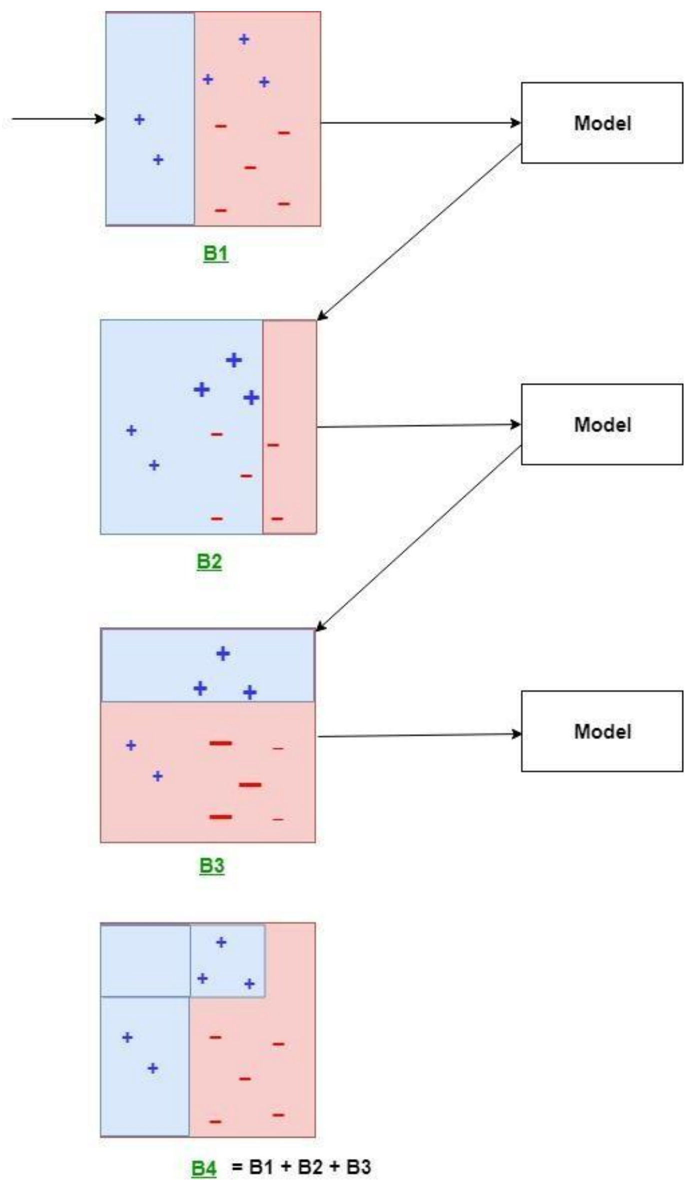


Figure 21 - Illustration Boosting classifier

Two popular types of boosting methods include:

- **Adaptive boosting or AdaBoost**: Yoav Freund and Robert Schapire are credited with the creation of the AdaBoost algorithm. This method operates iteratively, identifying misclassified data points and adjusting their weights to minimize the training error. The model continues optimize in a sequential fashion until it yields the strongest predictor.
- **Gradient boosting**: Building on the work of Leo Breiman, Jerome H. Friedman developed gradient boosting, which works by sequentially adding predictors to an ensemble with each one correcting for the errors of its predecessor. However, instead of changing weights of data points like AdaBoost, the gradient boosting trains on the residual errors of the previous predictor. The name, gradient boosting, is used since it combines the gradient descent algorithm and boosting method.

Adaptive Boosting

1) Adaptive Boosting model with base/default parameters

```
Accuracy of base ADB model on training data is 0.8463713477851084
Accuracy of base ADB model on testing data is 0.8135964912280702
```

On the base ADA Boost model, we can see a high accuracy on training data of 0.84 and test data of 0.81 which means there is no over fitting or under fitting in the data

2) Adaptive boosting model with tuned hyperparameters

We will now use hyperparameter tuning to find the best parameters for our ADA Boost model. The more important parameter in AdaBoost is n_estimators, with the addition of something called 'learning rate' which is an important parameter for boosting model algorithms given how boosting algorithms work as that will determine how each successive model will learn from the previous one. N_estimators we will use slightly different from our RF model.

Best parameters after grid search

```
{'learning_rate': 0.5, 'n_estimators': 25}
```

Accuracy scores

```
Accuracy of best ADB model on training data is 0.8369462770970783
Accuracy of best ADB model on testing data is 0.8157894736842105
```

Here we see a slightly dip in the accuracy score of the training data that is it reached to 0.83 which is almost 1 % lesser than the default parameters but it has also improved the test results slightly by a margin of 0.2% approximately.

Gradient Boosting

1) Gradient Boosting model with base/default parameters

```
Accuracy score on base Gradient boosting model on train data is 0.8925541941564562
Accuracy score on base Gradient boosting model on test data is 0.8355263157894737
```

Here we find a very good accuracy score of 89% for training and 83% for the testing data we don't see any overfitting or under fitting in this perhaps we have to see does it provide such good result after tuning as well.

2) Gradient boosting model with tuned hyperparameters

We will now use hyperparameter tuning to find the best parameters for our Gradient Boost model. The more important parameters in GradBoost is similar to RF model such as n_estimators, max_depth, max_features, min_sample_split, min_sample_leaf with the addition of something called 'learning rate' which is a important parameter for boosting model algorithms given how boosting algorithms work as that will determine how each successive model will learn from the previous one. We will narrow the parameters basis what we got in the RF model grid search, and keep only max_depth, n_estimators and learning_rate

Best parameters after grid search

```
{'learning_rate': 0.05, 'max_depth': 4, 'n_estimators': 71}
```

Accuracy scores

```
Accuracy score on best Gradient boosting model on train data is 0.8916116870876531
Accuracy score on best Gradient boosting model on test data is 0.8201754385964912
```

This model has performed significantly well even after tuning and there is no underfitting or over fitting in the model as the gap between training and testing of data is less than 10%

For detailed evaluation metrics of the boosting and bagging models, please refer to Q1.7

1.7 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUCscore for each model Final Model - Compare all models on the basis ofthe performance metrics in a structured tabular manner. Describe on which model is best/optimized

Model evaluation
Evaluation of various machine learning models we have built- Performance measurement of ML algorithms is judged by confusion matrix which comprise the classification count values of actual and predicted labels. The confusion matrix for binary classification is given by:

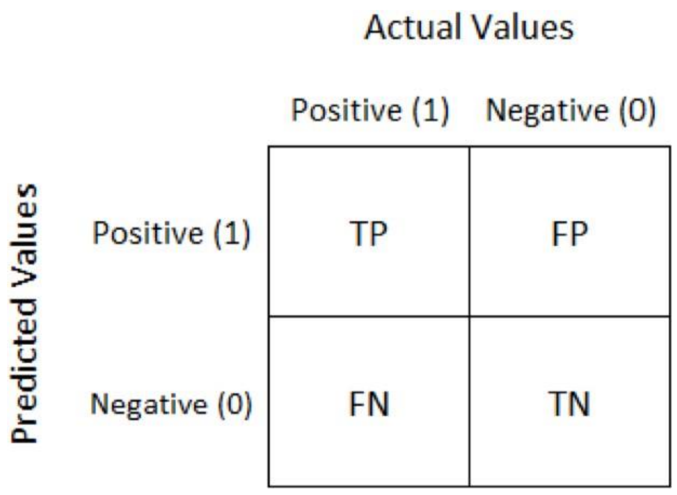


Figure 22 - Illustration Confusion Matrix

Confusion matrix cells are populated by the terms:

- True Positive (TP) - The values which are predicted as True and are actually True.
- True Negative (TN) - The values which are predicted as False and are actually False.
- False Positive (FP) - The values which are predicted as True but are actually False.
- False Negative (FN) - The values which are predicted as False but are actually True.

Classification performance metrics are based on confusion matrix values. The most popularly used metrics are;

Precision- measure of correctness achieved in prediction.

$$precision = \frac{TP}{TP + FP}$$

Recall (sensitivity)- measure of completeness, actual observations which are predicted correctly.

$$recall = \frac{TP}{TP + FN}$$

Specificity- measure of how many observations of false category predicted correctly.

$$specificity = \frac{TN}{TN + FP}$$

F1-Score- a way to combine precision and recall metric in a single term. F1score is defined as harmonic mean of precision and recall.

$$F1score = \frac{2 * precision * recall}{precision + recall}$$

ROC Curve- Receiver Operating Characteristic(ROC) measures the performance of models by evaluating the trade-offs between sensitivity (true positive rate) and false (1-specificity) or false positive rate.

AUC - The area under curve (AUC) is another measure for classification models is based on ROC. It is the measure of accuracy judged by the area under the curve for ROC.

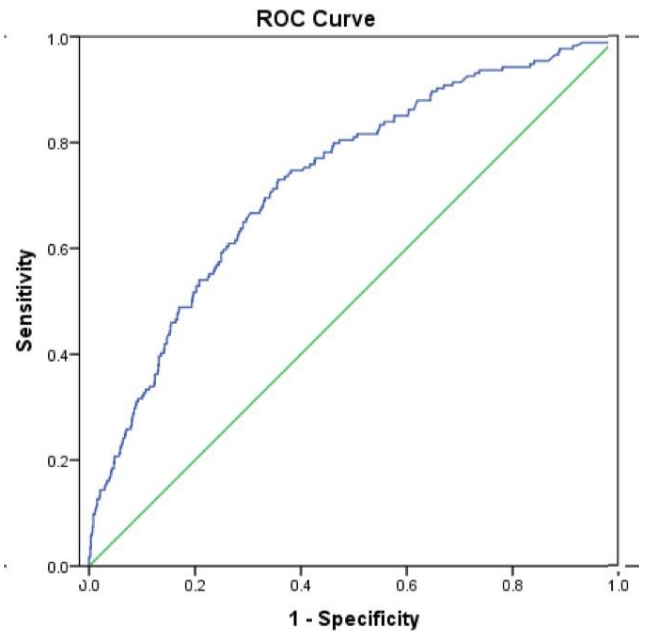


Figure 23 - Illustration ROC curve

Logistic Regression model evaluation

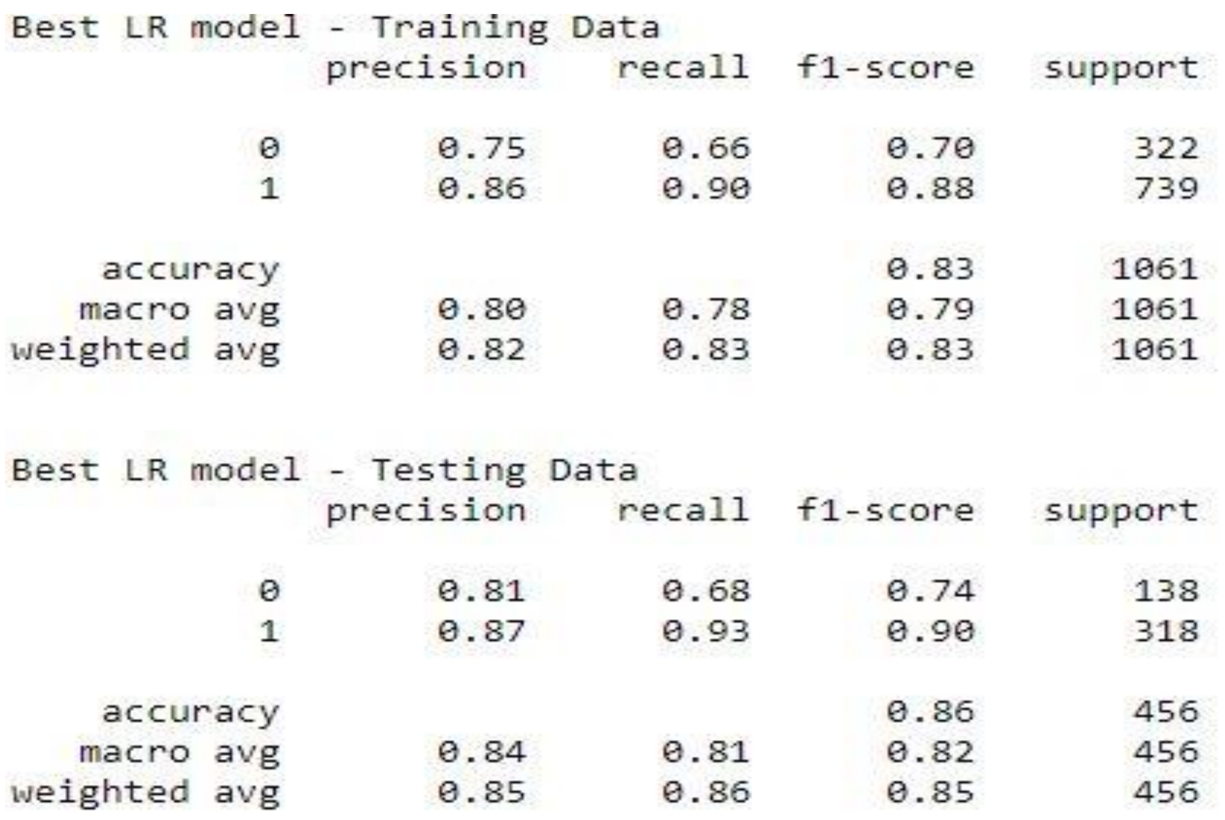
- Best model after dropping Gender and economic condition household

Accuracy scores

```
Accuracy Score - Best Model
Training: 0.8284637134778511
Testing:  0.8552631578947368
```

- Best LR model has the highest accuracy on test data of all logistic regression models

Classification report


```
Best LR model - Training Data
              precision    recall  f1-score   support

           0       0.75      0.66      0.70       322
           1       0.86      0.90      0.88       739

    accuracy                           0.83      1061
   macro avg       0.80      0.78      0.79      1061
weighted avg       0.82      0.83      0.83      1061


Best LR model - Testing Data
              precision    recall  f1-score   support

           0       0.81      0.68      0.74       138
           1       0.87      0.93      0.90       318

    accuracy                           0.86       456
   macro avg       0.84      0.81      0.82       456
weighted avg       0.85      0.86      0.85       456
```

- In this classification report we see a slightly dip in the recall value for the 0’s in comaprision to 1’s this is because of the imbalance in the data
- The accuracy score of the final Regression model is 83% for the training data and 86% for the test data which is quite high but other models have performed even better from this
- Precision is very good for both 0s and 1s along with a good overall accuracy scores which meansmost of the predictions the models make, turn out to be correct more than ~80% of the time

ROC curve and Area under the curve

Logistic regression model – train & test data

```
Best LR model train data AUC: 0.87691
```

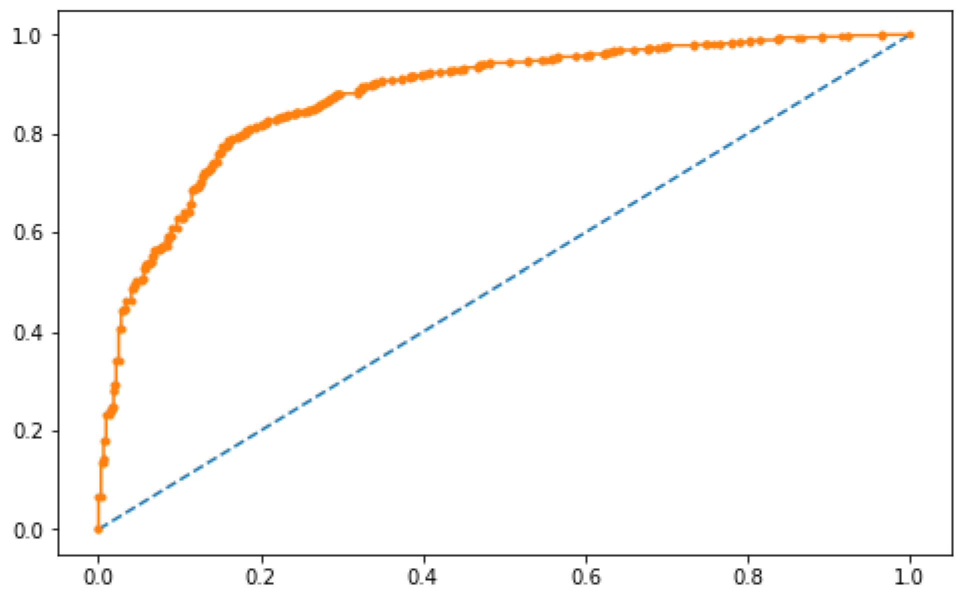


Figure 24 - ROC curve: LR (train)

```
Best LR model test data AUC: 0.91230
```

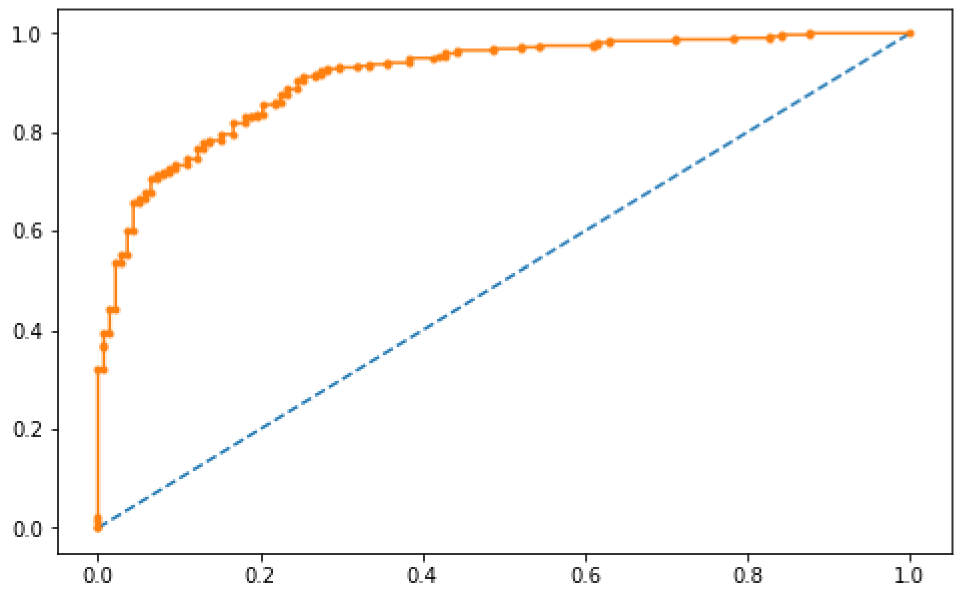


Figure 25 - ROC curve: LR (test)

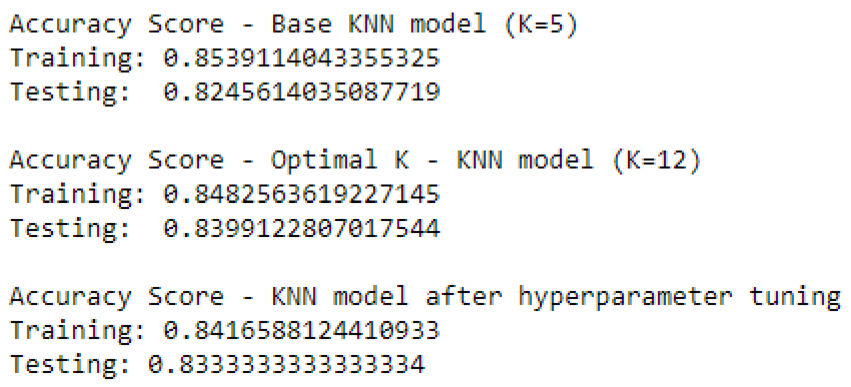
- AUC values - Train Data
    - Best LR model: 0.877
- AUC values - Test Data
    - Best LR model: 0.912

The AUC values are good for logistic regression model with the train & test data

KNN model evaluation

- Base KNN model (K=5)
- KNN model with optimal value of K=12
- KNN model after hyperparameter tuning

Accuracy scores



```
Accuracy Score - Base KNN model (K=5)
Training: 0.8539114043355325
Testing:  0.8245614035087719

Accuracy Score - Optimal K - KNN model (K=12)
Training: 0.8482563619227145
Testing:  0.8399122807017544

Accuracy Score - KNN model after hyperparameter tuning
Training: 0.8416588124410933
Testing: 0.8333333333333334
```

We see the best train data accuracy for the base KNN model (K=5) and best test data accuracy on the KNN model with optimal K (K=12). However, accuracy scores are very similar for all KNN models and slightly better than logistic regression models

Confusion matrix and classification report

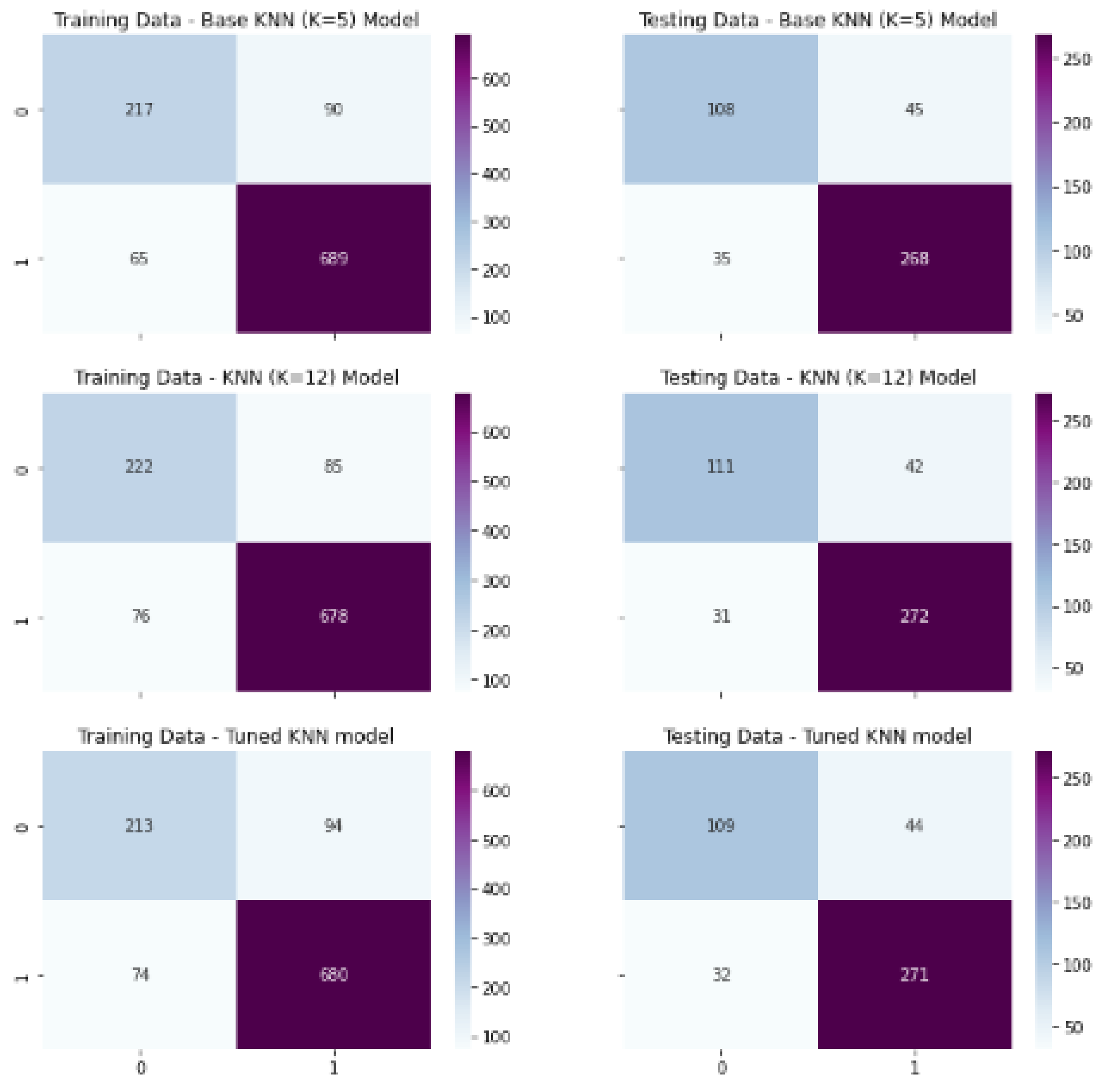


Figure 26 - KNN algorithm: Confusion matrix

```
Base KNN model (K=5) - Training Data
              precision    recall  f1-score   support

           0       0.77      0.71      0.74       307
           1       0.88      0.91      0.90       754

    accuracy                           0.85      1061
   macro avg       0.83      0.81      0.82      1061
weighted avg       0.85      0.85      0.85      1061


Base KNN model (K=5) - Testing Data
              precision    recall  f1-score   support

           0       0.76      0.71      0.73       153
           1       0.86      0.88      0.87       303

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

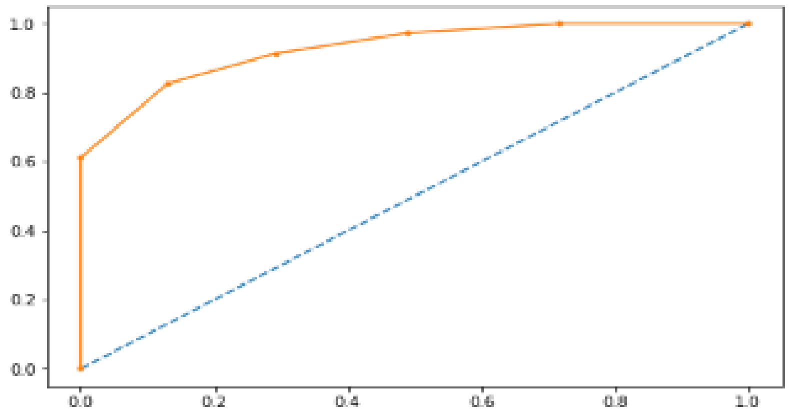
```
 Optimal K KNN model (K=12) - Training Data
              precision    recall  f1-score   support

           0       0.74      0.72      0.73       307
           1       0.89      0.90      0.89       754

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


Optimal K KNN model (K=12) - Testing Data
              precision    recall  f1-score   support

           0       0.78      0.73      0.75       153
           1       0.87      0.90      0.88       303

    accuracy                           0.84       456
   macro avg       0.82      0.81      0.82       456
weighted avg       0.84      0.84      0.84       456


Tuned KNN model - Training Data
              precision    recall  f1-score   support

           0       0.74      0.69      0.72       307
           1       0.88      0.90      0.89       754

    accuracy                           0.84      1061
   macro avg       0.81      0.80      0.80      1061
weighted avg       0.84      0.84      0.84      1061


Tuned KNN model - Testing Data
              precision    recall  f1-score   support

           0       0.77      0.71      0.74       153
           1       0.86      0.89      0.88       303

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

- While the accuracy scores are very similar, the recall values are different for the 3 KNN models with the tuned model having the worst recall values and the KNN model having the best recall value when k = 12.
- However, recall is still quite low again ranging between 0.6 to 0.7 with the recall touching 0.73 for the tuned KNN model. The specificity values are very high similar to logistic regression which means even the KNN models excellent are predicting 0s correctly but not 1s so much. The accuracy scores are similar to logistic regression as well but recall values are slightly better especially for the KNN models with K=5 and K=12
- Precision values are much higher for the KNN models as well compared to logistic regression showing how true most of the model's predictions eventually turn out to be

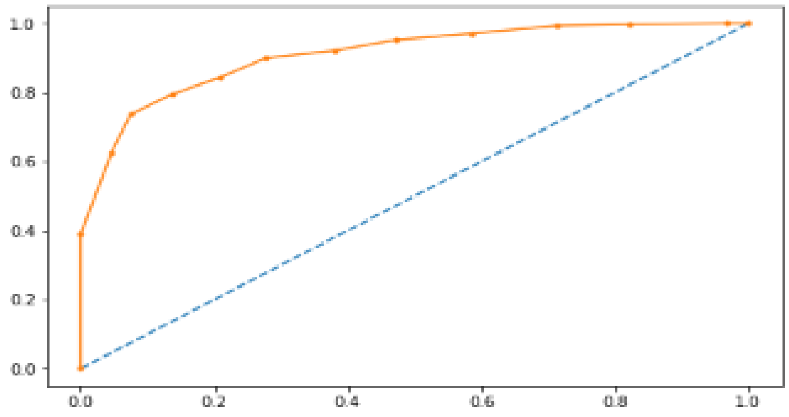
ROC curve and Area under the curve

Train data

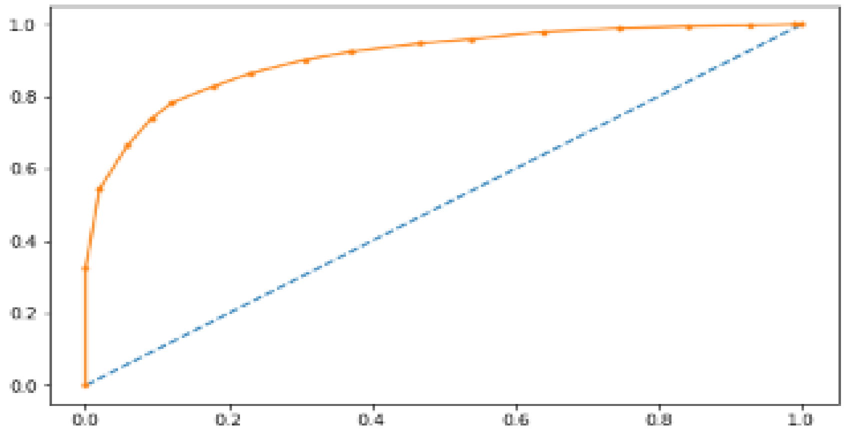
```
Base KNN model train data AUC: 0.92788
```



```
KNN (K=12) model train data AUC: 0.90949
```

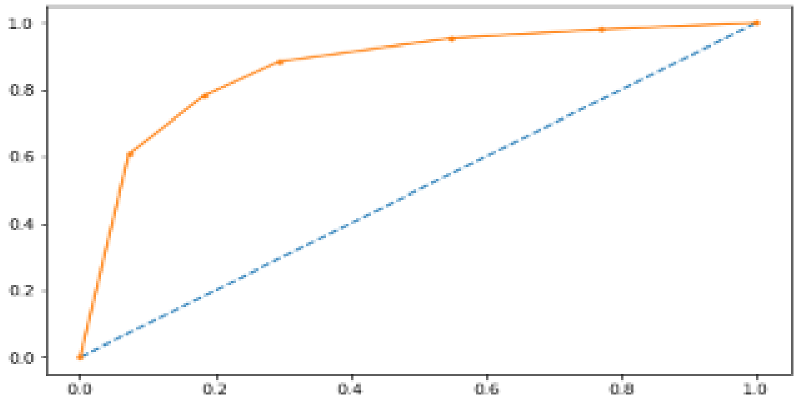


```
Tuned KNN model train data AUC: 0.90742
```

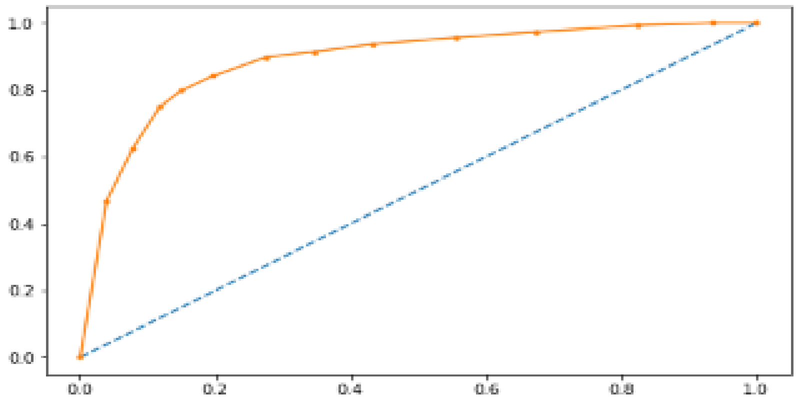


Test data

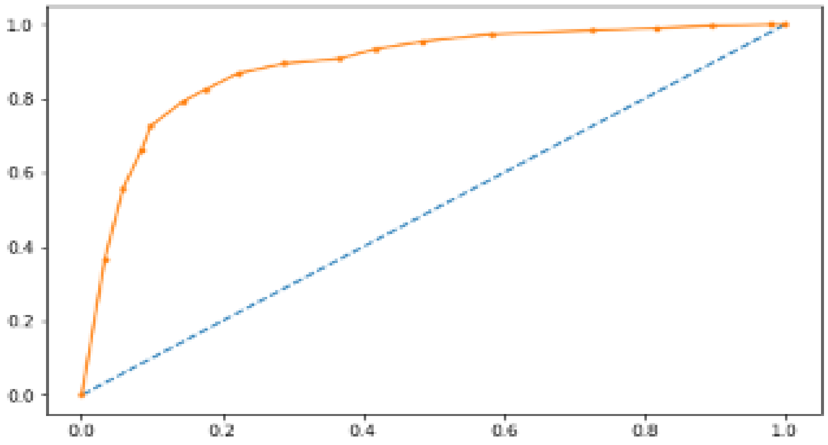
```
Base KNN model test data AUC: 0.86729
```



```
KNN (K=12) model test data AUC: 0.88762
```



```
Tuned KNN model test data AUC: 0.88886
```



In Roc AUC curve the knn model shows 91% area for train data and test data shows 87% coverage while in LR model it is vice versa the training data shows 87% area covered and test data shows 91% in it.

➢ Bagging classifier – Random Forest model evaluation

Accuracy scores

```
Accuracy Score - Base RF model
Training: 1.0
Testing:  0.8289473684210527

Accuracy Score - Tuned RF model
Training: 0.8689915174363808
Testing:  0.831140350877193
```

We see massive overfitting on the base RF model. The tuned RF model has solved this problem of overfitting in the data with also increasing the accuracy score in the test data.

Confusion matrix and classification report

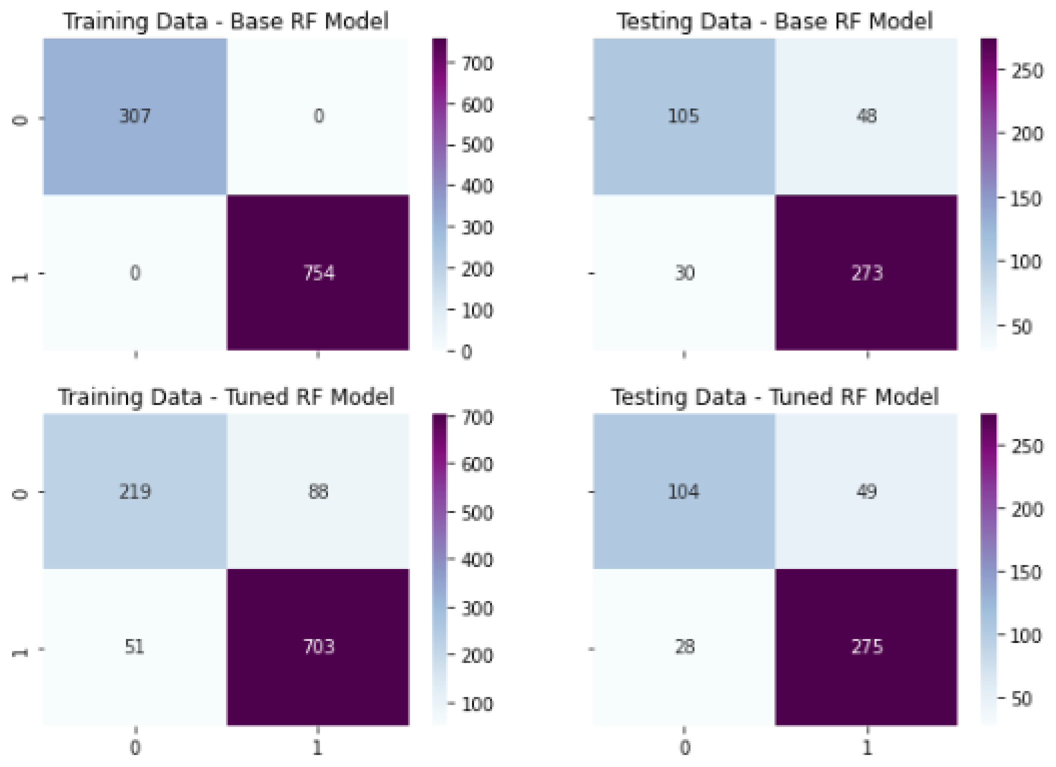


Figure 33 - Random Forest: Confusion matrix

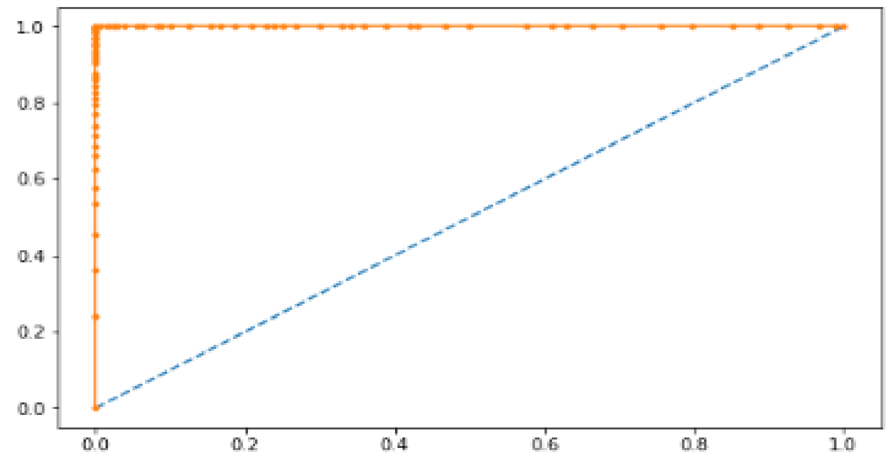
```
Base RF model - Training Data
              precision    recall  f1-score   support

           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


Base RF model - Testing Data
              precision    recall  f1-score   support

           0       0.78      0.69      0.73       153
           1       0.85      0.90      0.88       303

    accuracy                           0.83       456
   macro avg       0.81      0.79      0.80       456
weighted avg       0.83      0.83      0.83       456


Tuned RF model - Training Data
              precision    recall  f1-score   support

           0       0.81      0.71      0.76       307
           1       0.89      0.93      0.91       754

    accuracy                           0.87      1061
   macro avg       0.85      0.82      0.83      1061
weighted avg       0.87      0.87      0.87      1061


Tuned RF model - Testing Data
              precision    recall  f1-score   support

           0       0.79      0.68      0.73       153
           1       0.85      0.91      0.88       303

    accuracy                           0.83       456
   macro avg       0.82      0.79      0.80       456
weighted avg       0.83      0.83      0.83       456
```

- The various metrics are different for the Random Forest models on train and test data. Again, the precision values and accuracy are pretty good with specificity being extremely high. RF models like the previous ones is also very good at predicting 1’s compared to0’s

ROC curve and Area under the curve (AUC)

Train data

```
Base RF model train data AUC: 1.00000
```



```
Tuned RF model train data AUC: 0.93159
```

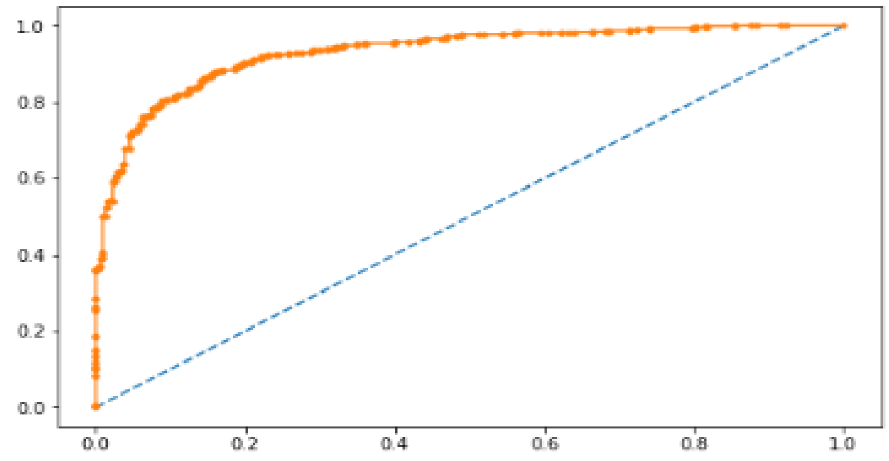
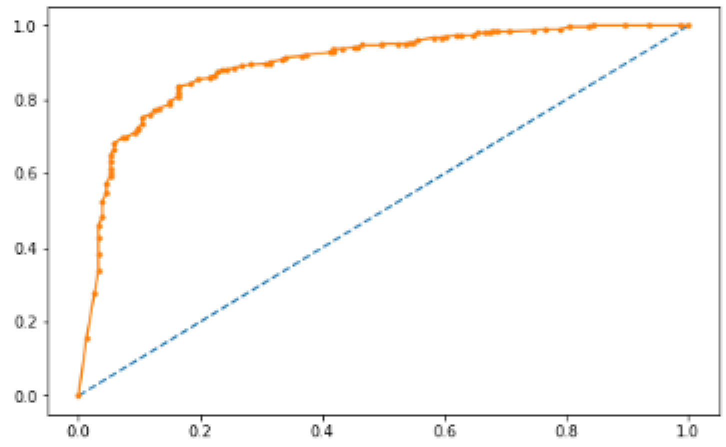


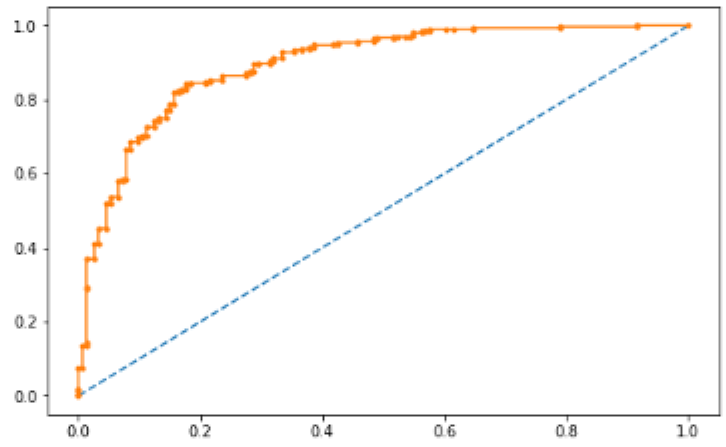
Figure 35 - ROC curve: RF (train)

Testing data

```
Base RF model test data AUC: 0.89486
```



```
Tuned RF model test data AUC: 0.89617
```



- AUC values - Train Data
    - Base RF model: 1.000
    - Tuned RF model: 0.932
- AUC values - Test Data
    - Base RF model: 0.895
    - Tuned RF model: 0.896

The AUC values on train & test data are significantly better for RF vs logistic regression and KNN models.

Boosting classifier – Adaptive Boosting model evaluation

Accuracy scores

```
Accuracy Score - Base ADB model
Training: 0.8463713477851084
Testing:  0.8135964912280702

Accuracy Score - Tuned ADB model
Training: 0.8369462770970783
Testing:  0.8157894736842105
```

As we see there is not much of a difference between the base and the tuned model in case of adaptive boosting method However, accuracy scores on the train & test data are lower for the ADA Boost algorithm compared to other models we have built.

Confusion matrix and classification report

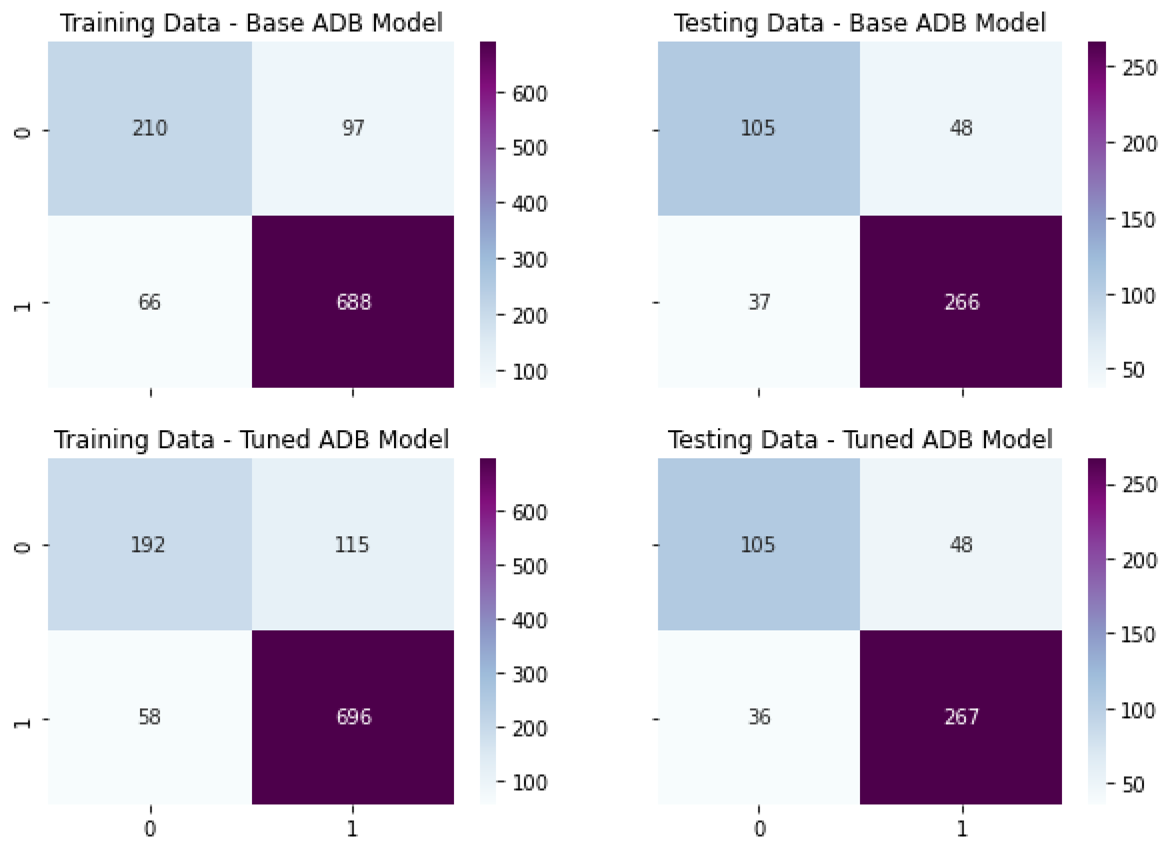


Figure 38 - Adaptive Boosting: Confusion matrix

```
Base ADB model - Training Data
              precision    recall  f1-score   support

           0       0.76      0.68      0.72       307
           1       0.88      0.91      0.89       754

    accuracy                           0.85      1061
   macro avg       0.82      0.80      0.81      1061
weighted avg       0.84      0.85      0.84      1061
```

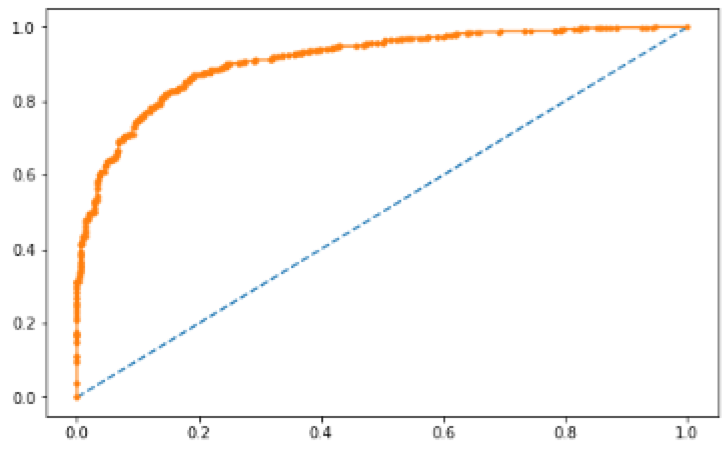
```
Base ADB model - Testing Data
              precision    recall  f1-score   support

           0       0.74      0.69      0.71       153
           1       0.85      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


Tuned ADB model - Training Data
              precision    recall  f1-score   support

           0       0.77      0.63      0.69       307
           1       0.86      0.92      0.89       754

    accuracy                           0.84      1061
   macro avg       0.81      0.77      0.79      1061
weighted avg       0.83      0.84      0.83      1061


Tuned ADB model - Testing Data
              precision    recall  f1-score   support

           0       0.74      0.69      0.71       153
           1       0.85      0.88      0.86       303

    accuracy                           0.82       456
   macro avg       0.80      0.78      0.79       456
weighted avg       0.81      0.82      0.81       456
```

- The various metrics are different for the Adaptive Boosting models on train and test data. Again, the precision values and accuracy are pretty good with specificity being extremely high especially on training data. ADB models like the previous ones is also very good at predicting 1’s compared to 0’s

ROC curve and Area under the curve

Train data

```
Base ADB model train data AUC: 0.91190
```



```
Tuned ADB model train data AUC: 0.89833
```

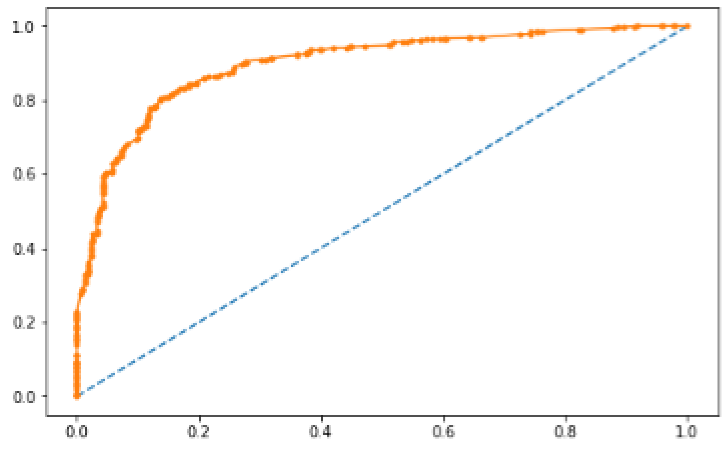
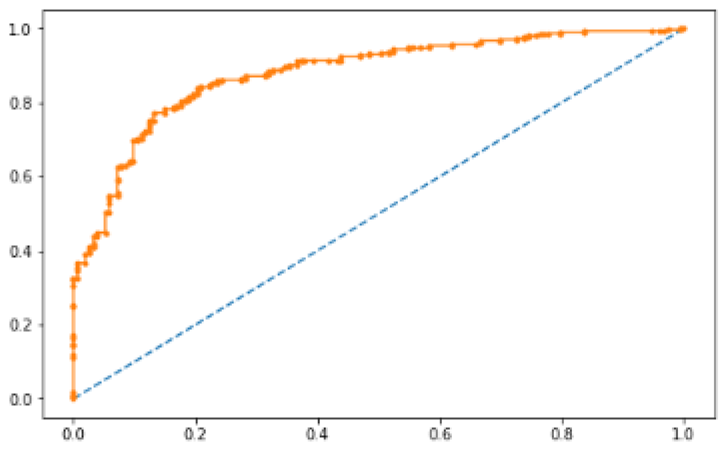


Figure 40 - ROC curve: ADB (train)

Test data

```
Base ADB model test data AUC: 0.88055
```



```
Tuned ADB model test data AUC: 0.88716
```

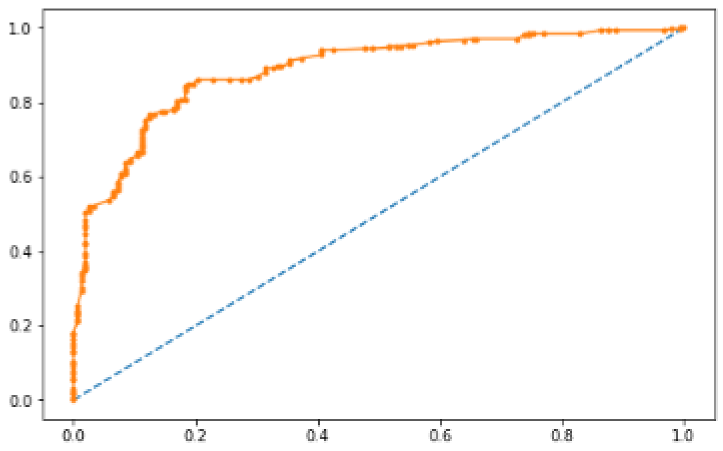


Figure 41 - ROC curve: Tuned ADB (test)

- AUC values - Train Data
  - Base ADB model: 0.912
  - Tuned ADB model: 0.898
- AUC values - Test Data
  - Base ADB model: 0.880
  - Tuned ADB model: 0.887

The AUC value for the tuned ADB model on train data is very good with a drop in values on train data fortuned ADA Boost models. The tuned ADB model also has a better AUC than the base ADB model on test data. However we see the ROC AUC value of RF model shows a higher area covered as compare to this.

Boosting classifier – Gradient Boosting model evaluation

Accuracy scores

```
Accuracy Score - Base Grad Boost model
Training: 0.8925541941564562
Testing:  0.8355263157894737

Accuracy Score - Tuned Grad Boost model
Training: 0.8916116870876531
Testing:  0.8201754385964912
```

Gradient Boosting has resulted in the best model when compared to all other algorithms. This model is having the best accuracy score when compared with all the other algorithms with 89% off accuracy in training data and 82% in test data which is a bit on lower side.

Confusion matrix and classification report

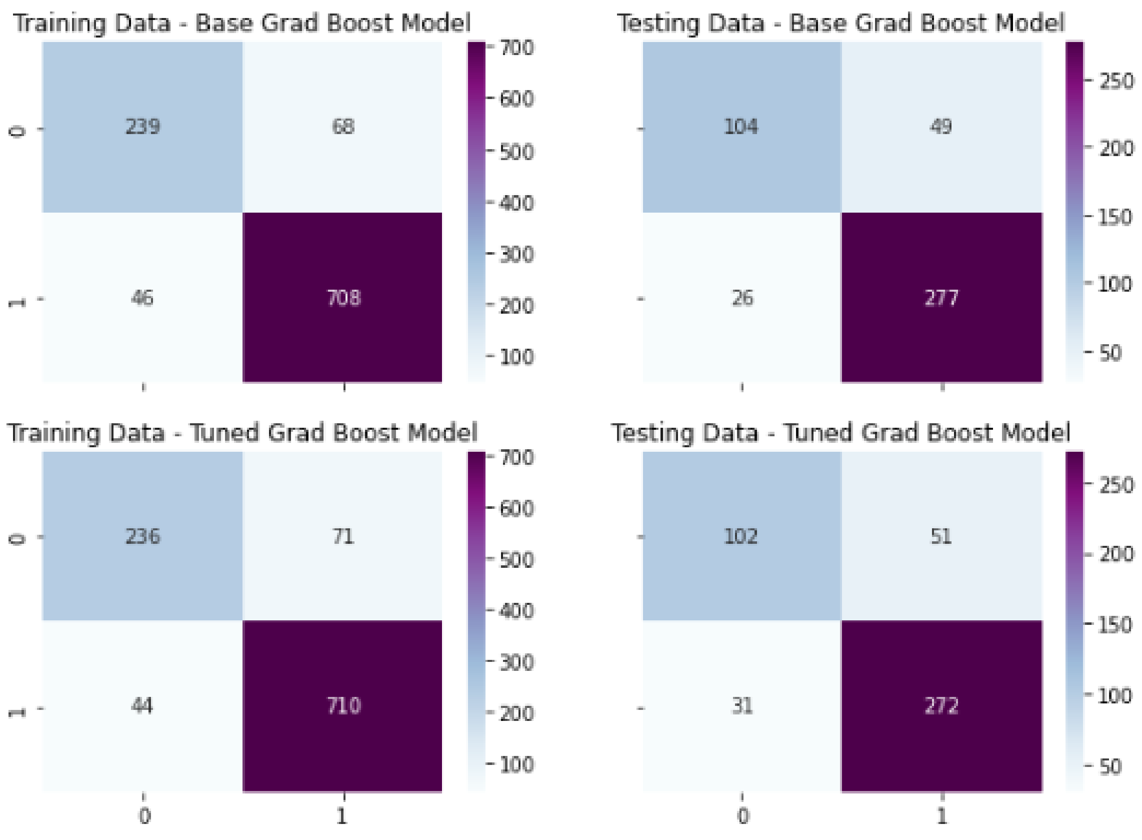


Figure 43 - Gradient boosting: Confusion matrix

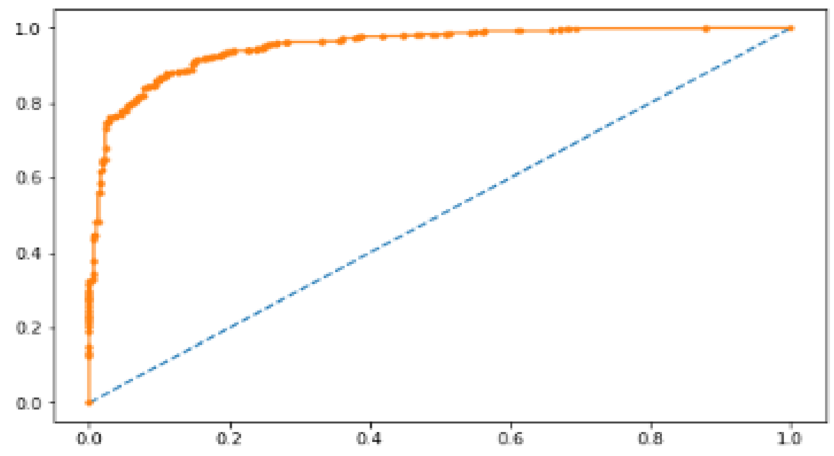
```
Base Grad Boost model - Training Data
              precision    recall  f1-score   support

           0       0.84      0.78      0.81       307
           1       0.91      0.94      0.93       754

    accuracy                           0.89      1061
   macro avg       0.88      0.86      0.87      1061
weighted avg       0.89      0.89      0.89      1061


Base Grad Boost model - Testing Data
              precision    recall  f1-score   support

           0       0.80      0.68      0.73       153
           1       0.85      0.91      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


Tuned Grad Boost model - 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


Tuned Grad Boost model - Testing Data
              precision    recall  f1-score   support

           0       0.77      0.67      0.71       153
           1       0.84      0.90      0.87       303

    accuracy                           0.82       456
   macro avg       0.80      0.78      0.79       456
weighted avg       0.82      0.82      0.82       456
```

- In this model as we see the recall value is also the highest with around 77% which we were unable to achieve in any other model
- Precision also is shows a decent result when it comes to 0s & 1s compared to other models .

ROC curve and Area under the curve

Train Data

```
Base Grad Boost model train data AUC: 0.95116
```



```
Tuned Grad Boost model train data AUC: 0.94880
```

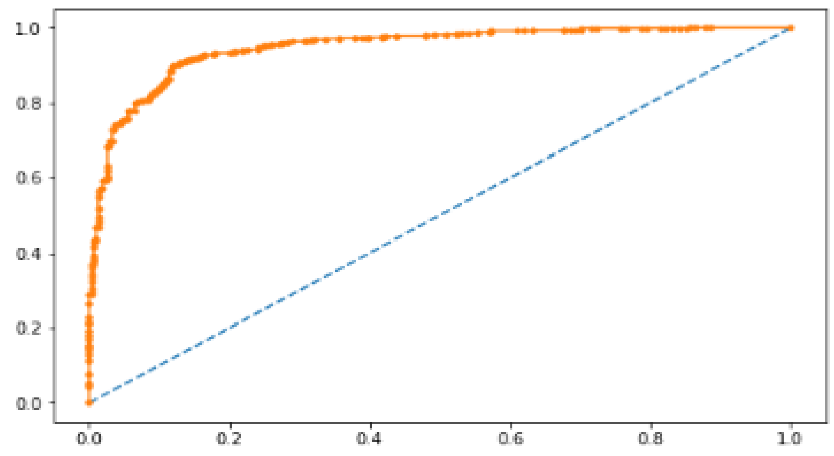
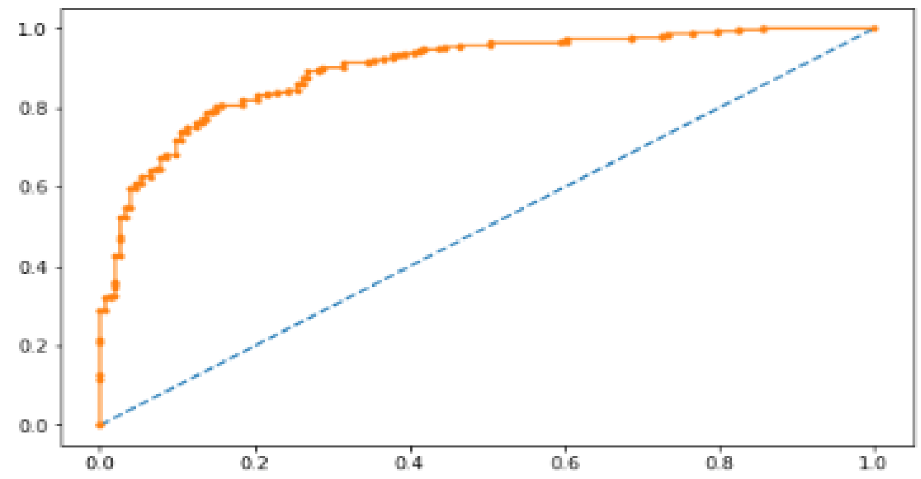


Figure 45 - ROC curve: Grad Boost (train)

Test data

```
Base Grad Boost model test data AUC: 0.89856
```



```
Tuned Grad Boost model test data AUC: 0.89579
```

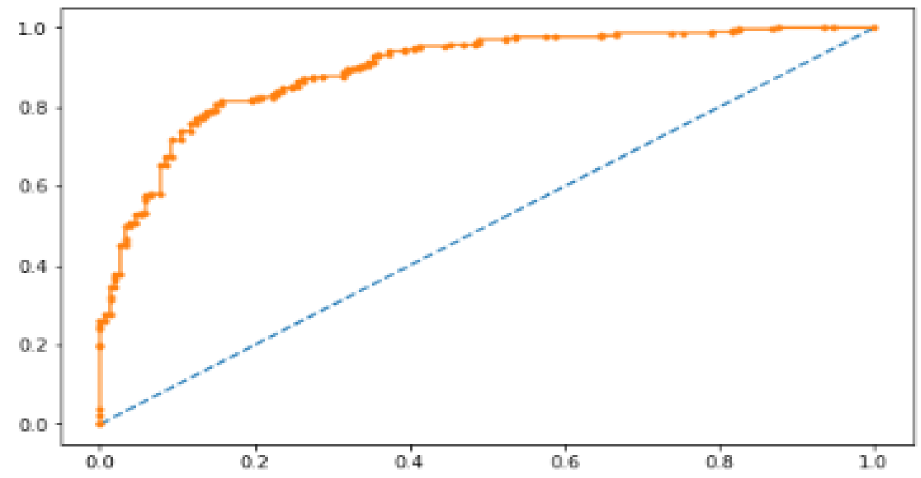


Figure 46 - ROC curve: Tuned Grad Boost (test)

- AUC values - Train Data
    - Base Grad Boost model: 0.951
    - Tuned Grad Boost model: 0.945
- AUC values - Test Data
    - Base Grad Boost model: 0.899
    - Tuned Grad Boost model: 0.896

In this ROC AUC we see a very huge area which is been covered under the curve and the test result as well show a very good result there is no underfitting or overfitting in this model.

Comparison of all machine learning models and evaluation metrics

We will compare the various evaluation metrics of the multiple models built so far to determine the best model for our dataset – we are removing the massively overfitted models and significantly underperforming models from this comparison for ease of reference.

Algorithm	Model	Training Data					Testing Data				
		Accuracy	Precision (1)	Recall (0)	Recall (1)	AUC	Accuracy	Precision (1)	Recall (0)	Recall (1)	AUC
Logistic Regression	Best	0.83	0.86	0.66	0.9	0.88	**0.86**	**0.87**	0.68	**0.93**	**0.91**
K - Nearest Neighnours	K = 5	0.85	0.88	0.71	0.91	0.93	0.82	0.86	0.71	0.91	0.87
	K = 12	0.85	0.89	0.72	0.90'	0.91	0.84	0.87	**0.73**	0.90'	0.89
	Tuned	0.84	0.88	0.69	0.90'	0.91	0.83	0.86	0.71	0.89	0.89
Random Forest	Tuned	0.87	0.89	0.71	0.93	0.93	0.83	0.85	0.68	0.91	0.90'
Adaptive Boosting	Tuned	0.84	0.86	0.63	0.92	0.90'	0.82	0.85	0.69	0.88	0.89
Gradient Boosting	Tuned	**0.89**	**0.91**	**0.77**	**0.94**	**0.95**	0.82	0.84	0.67	0.90'	0.90'

- From the above comparison table, we can see the tuned Gradient Boosting model being the best performing model when it comes to training data whereas the Logistic Regression is the best models when it comes to testing data

- Across the board, we see a good accuracy score ~0.83-0.95 with specificity (recall for 1's) being extremely high (0.9+) with recall (0's) being low (0.6-0.7) with some exceptions. KNN model is performing significantly better on recall (0.71) compared to all other models. Overall models are excellent at predicting 1's (Labour Party) but not so good at predicting 0's(Conservative Party) correctly

- Precision is also quite high (0.85+ in most cases) which means that most of the model'spredictions eventually turn out to be correct too

- AUC scores are fairly good across all models almost similar therefore it was very difficult to chose any particular model

- **Due to no overfitting/underfitting and significantly better performance on train data, I would recommend the Gradient Boosting Model as the best model for our case study. Random Forest & Logistic Regression is also a good model for this dataset**

1.8 Based on these predictions, what are the insights?

EDA insights

1.8.1 With EDA we can clearly identify from this data that the labour party has a brighter future and has more number of supporters

1.8.2 Despite of the age of voters, their political knowledge, their economic condition, gender every area have a marginal difference between the voters of both the parties and every where we can see a clear winner for the labour party

1.8.3 We saw that in various models as well that many of the columns stood to be very important but major chunk of voters were in support of labour party only

1.8.4 In assesments of both the leaders that is in blair and hague, in blair there were more supporters for leader of labour party and in hague there were more supporters for leader of conservative party but with a narrow margin

1.8.5 Europe was a good column for predicting the sentiments of the people towards both the parties

1.8.6 The age of people do not have much of a difference in the voting partern because as we see every age group was preferring labour party over conservative party

Machine learning model insights

1.8.7 Across the board, we see a good accuracy score ~0.83-0.95 with specificity (recall for 1's) being extremely high (0.9+) with recall (0's) being low (0.6-0.7) with some exceptions. KNN model is performing significantly better on recall (0.71) compared to all other models. Overall models are excellent at predicting 1's (Labour Party) but not so good at predicting 0's(Conservative Party) correctly

1.8.8 In business insights as we see in the EDA as well in the model buildings conservative party is loosing with a huge margins so if it needs to improve its position it needs to work more hard towards the state needs to build more good relation with their voters try to build a better brand image of the leader of conservative party in front of the people so that the voters start liking his leader

1.8.9 And from Labour party point of view it is already doing well in their area as there are only two parties and labour party leader is more preferred by the voters it just needs to ingage more with the voters try to meet the hague people who are more with the favour of opposite party leader and try to understand about their concerns in order to turn them on our side.

1.8.10 Both the leaders also have to promote the work they are doing in the public so that people are more aware about the work that they are being doing so that the voters can trust them even more

1.8.11 For conservative party it is a difficult journey to retain back the voters but it is not an impossible with constant efforts, time management and gaging more with the people it can be easily achieved