Report- Machine Learning

Tanushri Das

Table of Contents:

Problem 1

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

1.1) Read the dataset. Do the descriptive statistics and do null value condition check.

Ans)

We have been provided with details of recent election survey conducted by CNBE news channel. The survey was conducted on 1525 voters and the attributes taken into account have been listed below with their description.

Sr.No	Variable Name	Description
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.

Table 1: Data Description

The purpose is to build different models to predict which party a voter will vote for on the basis of the given information, and create an exit poll that will help in predicting overall win and seats covered by a particular party.

	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

Table 2: First five rows of dataset

From the table 1 we can understand the data distribution of each column of the dataset.

```
<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
```

Figure 1: Information about dataset

Apart from vote and gender, all other columns are numerical in nature.

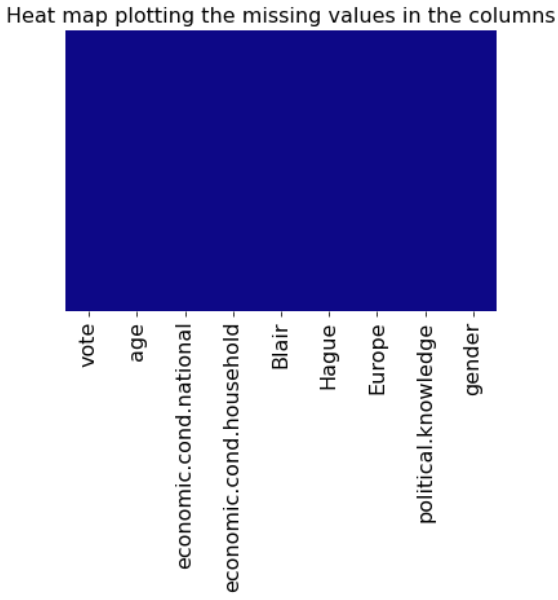


Figure 2: Plotting null values

There are no null values in the dataset.

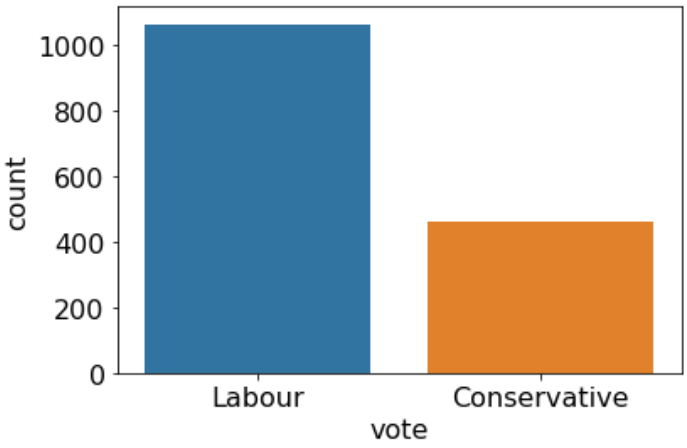


Figure 3: Distribution of Target column

We can see that from the given survey, majority of voters have voted for Labour party. 1057 (70%) people have voted for labour party and 460 (30 %) people have voted for conservative party, which implies this is an imbalanced dataset.

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
count	1525	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000	1525.000000	1525
unique	2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2
top	Labour	NaN	NaN	NaN	NaN	NaN	NaN	NaN	female
freq	1063	NaN	NaN	NaN	NaN	NaN	NaN	NaN	812
mean	NaN	54.182295	3.245902	3.140328	3.334426	2.746885	6.728525	1.542295	NaN
std	NaN	15.711209	0.880969	0.929951	1.174824	1.230703	3.297538	1.083315	NaN
min	NaN	24.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	NaN
25%	NaN	41.000000	3.000000	3.000000	2.000000	2.000000	4.000000	0.000000	NaN
50%	NaN	53.000000	3.000000	3.000000	4.000000	2.000000	6.000000	2.000000	NaN
75%	NaN	67.000000	4.000000	4.000000	4.000000	4.000000	10.000000	2.000000	NaN
max	NaN	93.000000	5.000000	5.000000	5.000000	5.000000	11.000000	3.000000	NaN

Table 3: Description of data

As we know the scales and values of all columns, the distribution is pretty clear in terms of ranges. The ages of people are from minimum 24 till maximum 93 years with mean age around 54 years which is around the median value. In other columns too the mean is similar to the median values. We can see that mean values for age, economic.cond.national, economic.cond.household, Blair, Hague and Europe are slightly more than median values which can imply there are equally more voters that have given higher rating than the median value in these cases and more people are older than 53 so they are right slightly right skewed whereas for political.knowledge the mean is slightly lesser than median which means it can be left skewed. More female voters were surveyed than male voters in this case study.

There were 8 duplicate records in this case. As they do not add any value they will be dropped

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

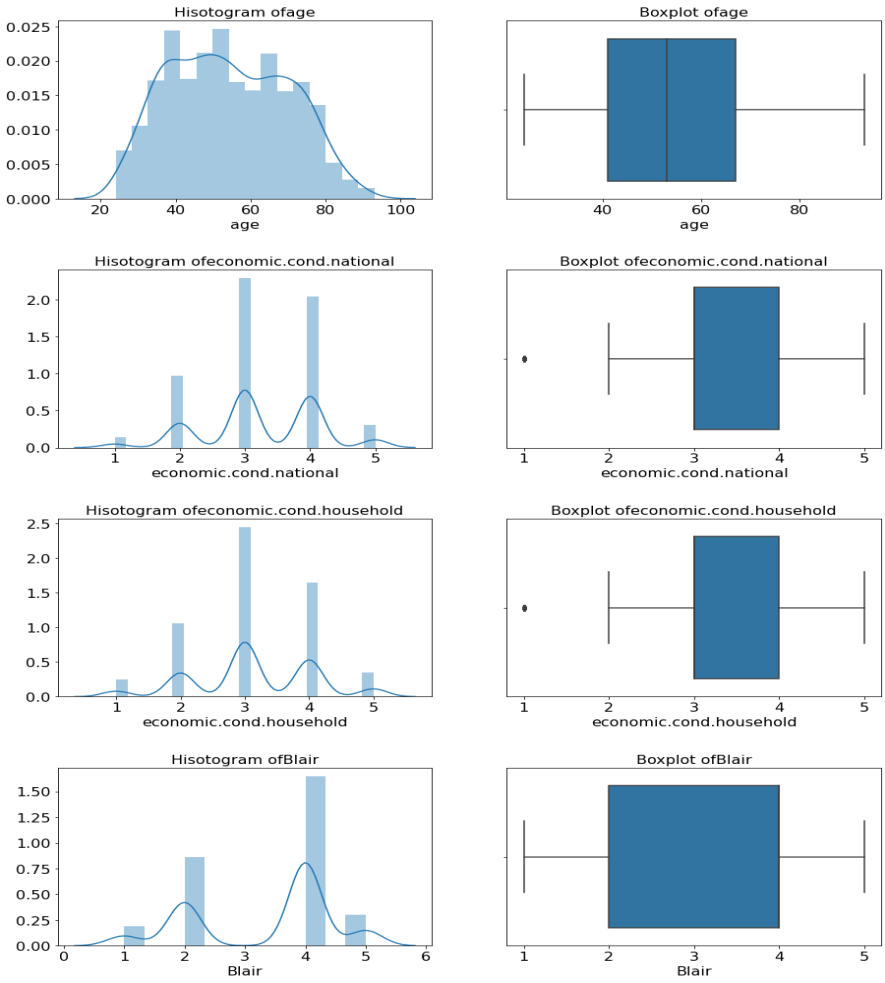
Ans)

Univariate Analysis:

Checking the individual value counts for each value in all columns from line [13] of notebook, we can see that many people who were surveyed are in their late 30s and 40s. The maximum scores given for each category are as follows:

- economic.cond.national: 3
- economic.cond.household:3
- Blair: 4
- Hague: 2
- Europe: 11
- political.knowledge: 2

Looking at the distribution of values individually,



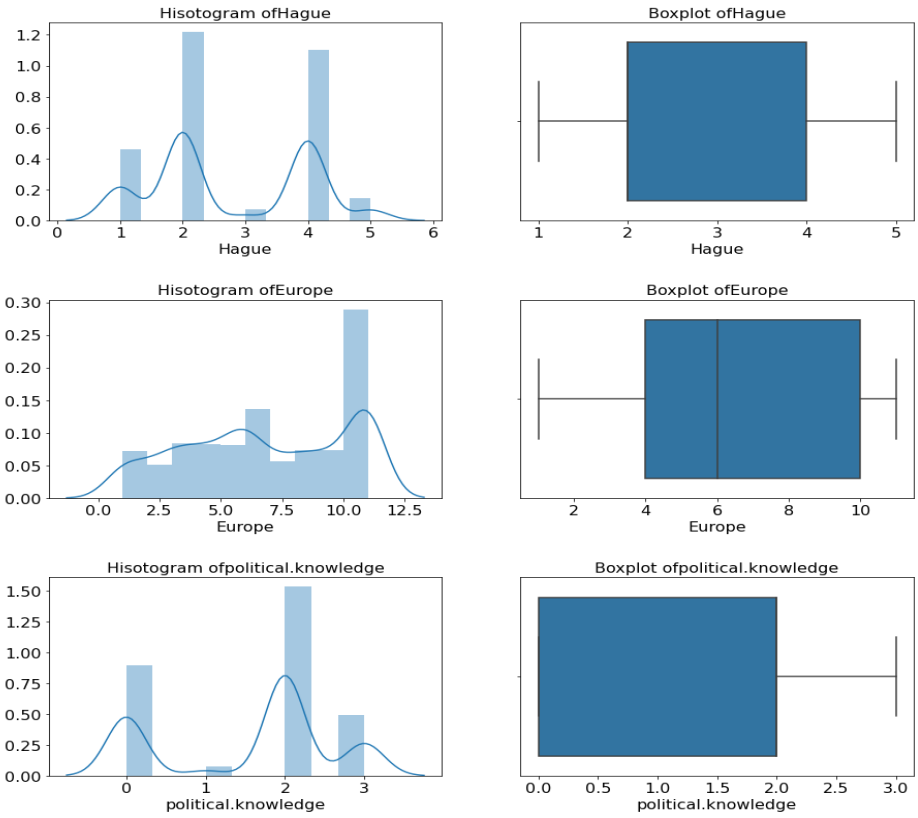


Figure 4: Distribution of all columns

Age seems to be normally distributed, slightly right skewed. Majority of people consider their current national and household economic conditions to be average, of around 3 and 4. Many people have assessed Blair with impressive high numbers of 4 who is also the Labour party leader which had secured maximum votes from the given survey. No one even scored him 3 which is average which shows that majority of people do consider him as a great leader. Whereas for Hague, people have a divided opinion about him. Majority of people have either given him high points of 4 or low points of 2 which nullified the average assessment to a score of 3.

Majority of people have given highest score to Europe which shows very high "Eurosceptic" sentiments among people. The voters considered their political knowledge to be fairly good with majority of them assessing themselves as a 2 on a scale of 3.

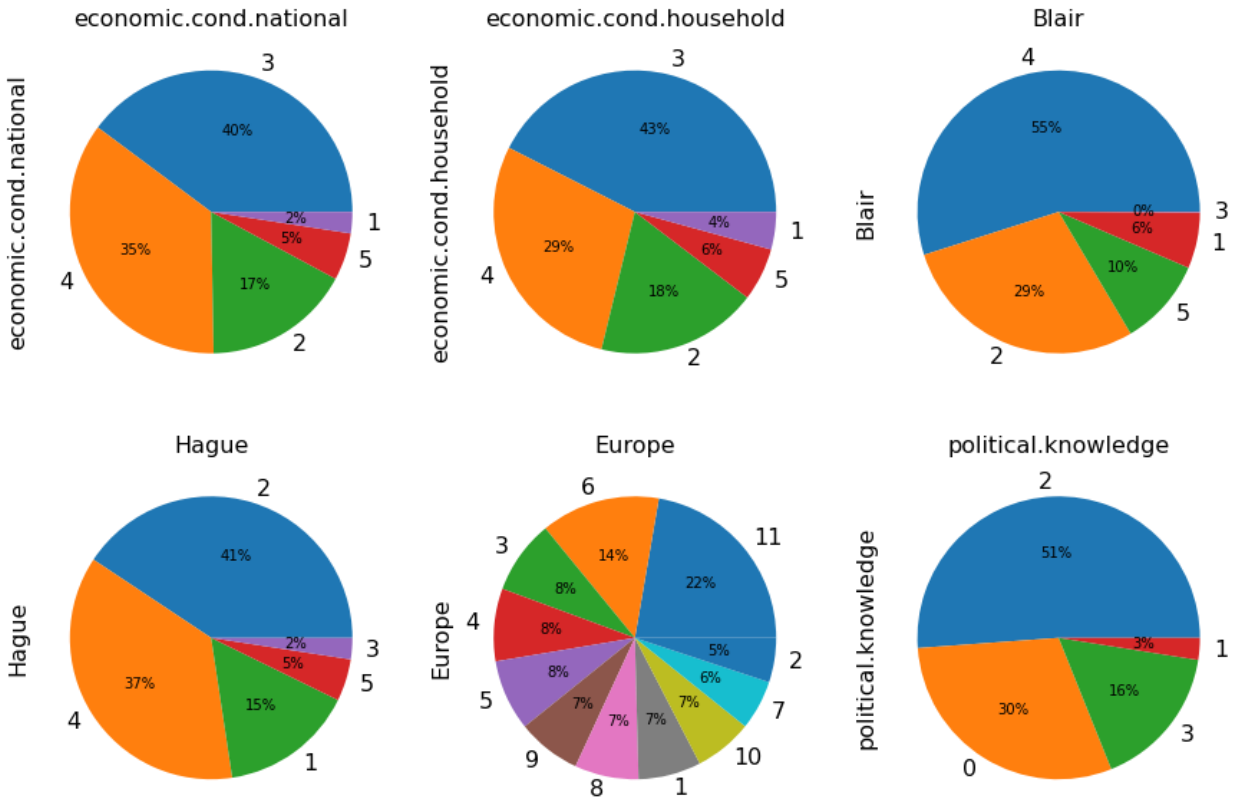


Figure 5: Distribution of assessment scores

We can view the distribution of assessment value for each one of the columns from above pie chart.

When we assess the party leader scores based on the votes, we get the following insights:

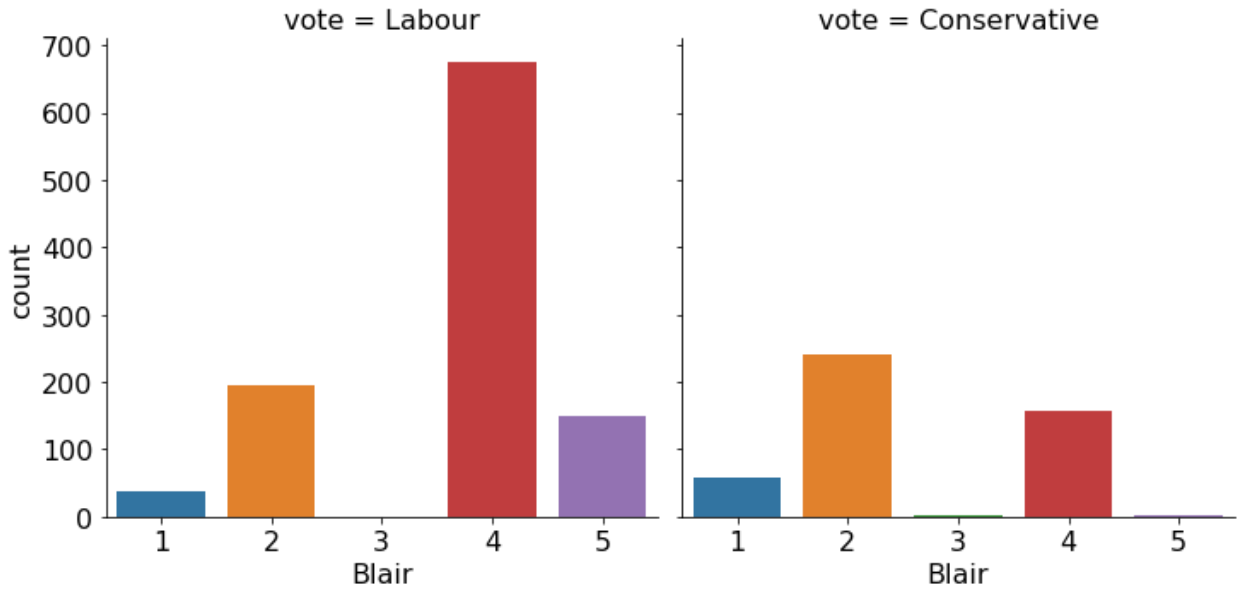


Figure 6: Blair's assessment scores based on party votes

The people who have voted for Labour party have scored pretty positively for Blair. We can see that quite a few opposition party voters have also assessed Blair with high scores of 4.

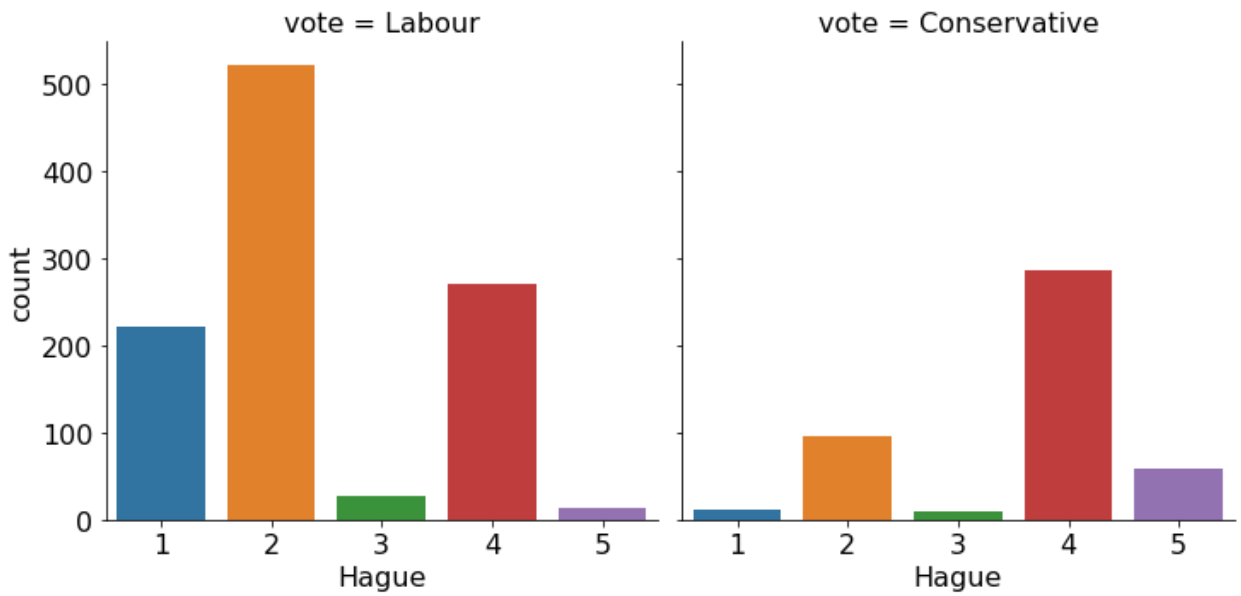


Figure 7: Hague's assessment scores based on party votes

Hague has pretty high assessment from people who have voted for Conservative party but seems to be pretty negatively assessed from the voters of Labour party.

To compare the scales of each column and check for presence of outliers, we will be looking at boxplots of all variables together.

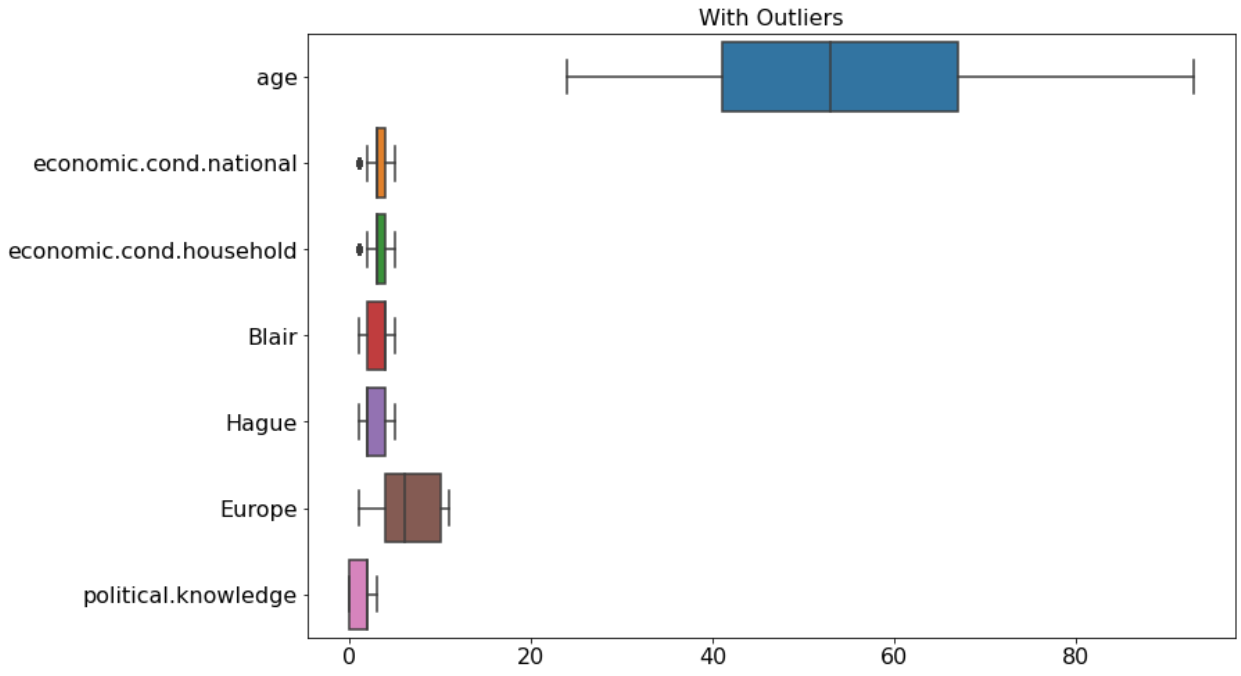


Figure 8: Checking for outliers

We can see that only 1 value in economic national and household condition is an outlier. But since this is a valid value so we will not be treating it in the case study.

Bivariate Analysis:

Since the columns are majorly ordinal/ categorical in nature, in order to check correlation between them we have used the Spearman and Kendall correlation too along with Pearson's correlation for comparison. We can see from line [21] that correlation coefficient values are pretty much same from all of them. So we'll be referring to Kendall's coefficient values for this case.

We can see that there is no much statistical correlation among variables among each other. The highest of Blair can be seen for votes which can be attributed because of the fact that majorities have voted for Labour party and Blair is its leader, and he has also received good assessments from his voters as we have seen from above insights.

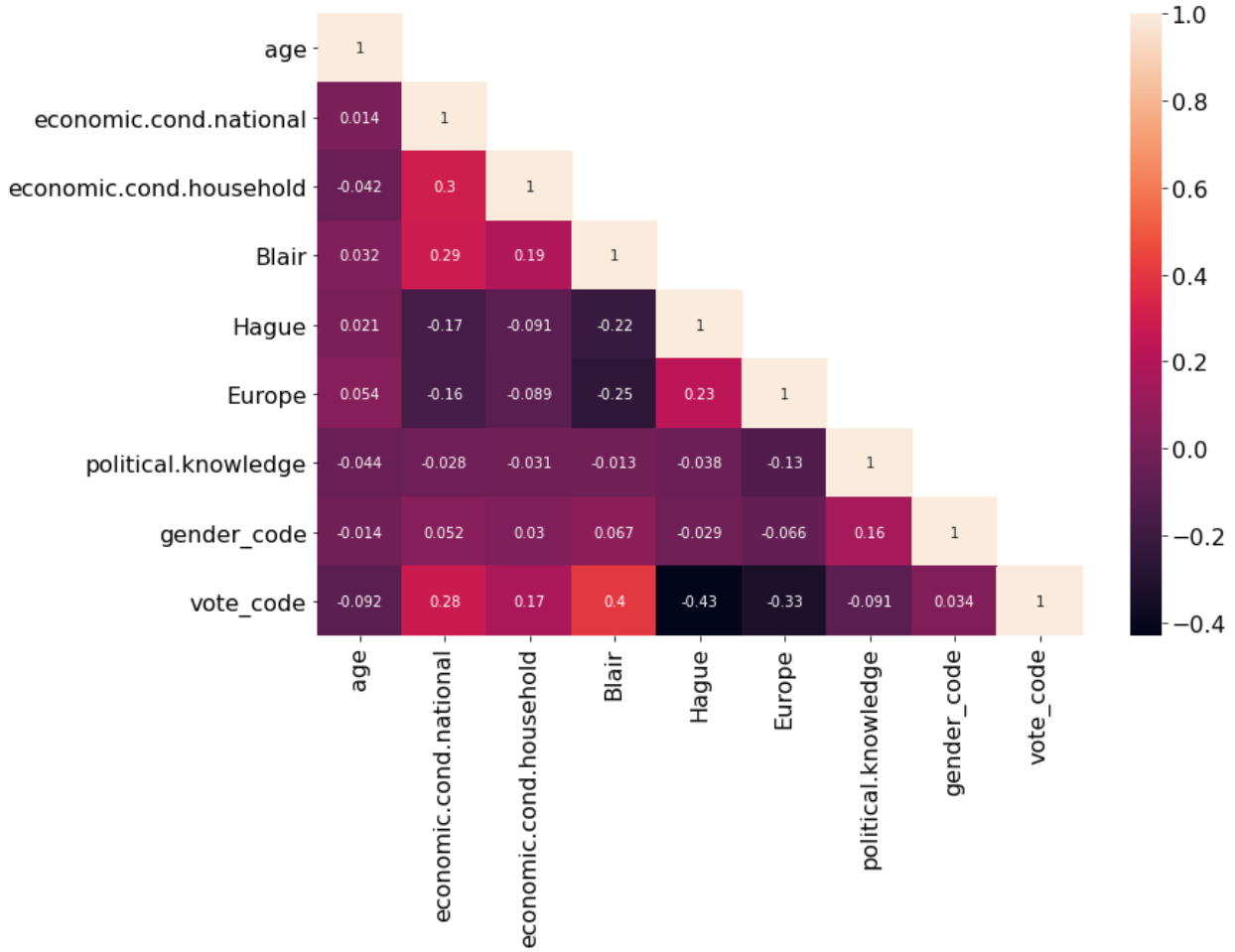


Figure 9: Correlation plot

To check if gender and political knowledge would have any impact on the votes we conducted statistical tests for them at line [24]. We can say with 95% confidence that gender has no significant impact to the votes but we cannot deny the significance of political knowledge of a voter to the party he/she will vote for.

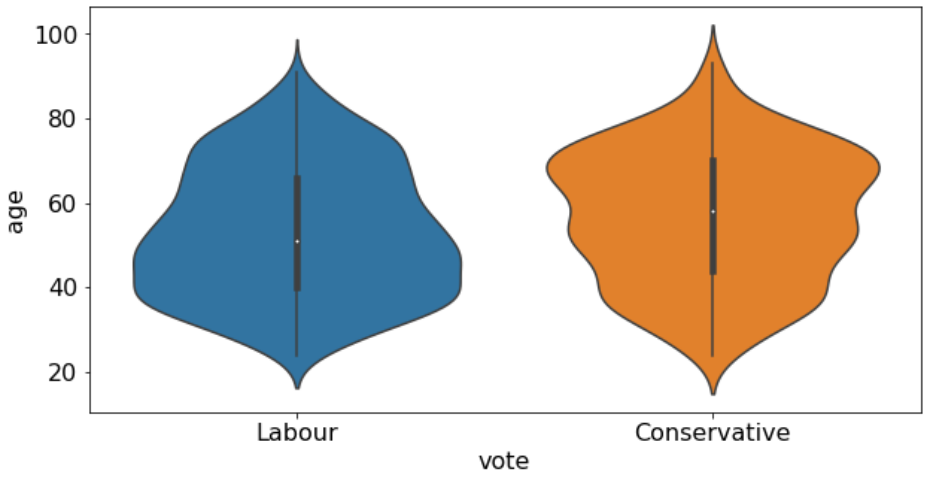


Figure 10: Vote VS age

We can see from above plot that maximum votes for Labour party are in the range of around 40-50 years and for Conservative party they are in higher bracket of age maximum around 60-70 years.

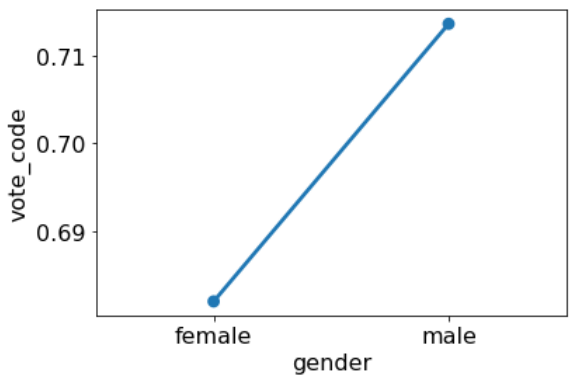


Figure 11: Vote VS gender

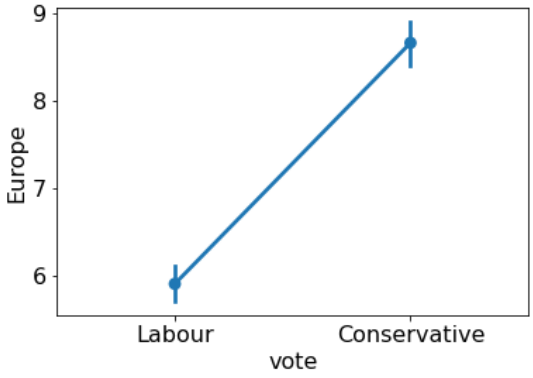


Figure 12: Vote VS Europe

From the above plots we can infer that men are slightly more likely to vote for Labour party as per the survey. And voters of Conservative party are likely to have higher “Eurosceptic” sentiments than voters of Labour party.

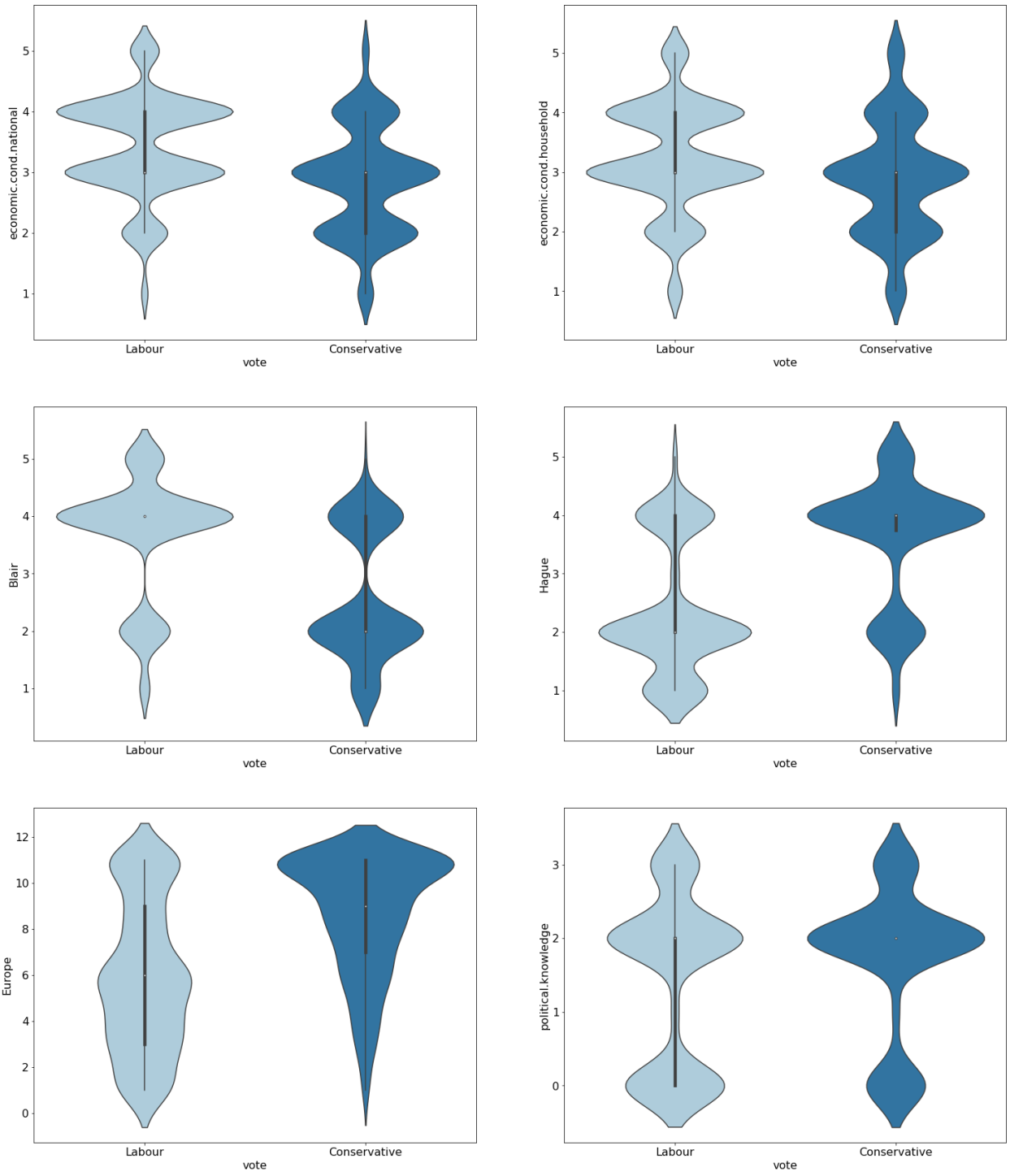


Figure 13: Violinplot for each column with vote

We can see some differences in trends on assessments scores by voters for the given two parties.

- **economic.cond.national:** Voters of Labour party consider the current national economic conditions to be better than voters of Conservative party
- **economic.cond.household:** Voters of Labour party have similar assessment of current household economic conditions to as voters of Conservative party
- **Blair:** Voters of Labour party have a highly positive opinion for Blair as compared to voters of Conservative party
- **Hague:** Voters of Conservative party have a highly positive opinion for Hague as compared to voters of Labour party
- **Europe:** Voters of Conservative party have high "Eurosceptic" sentiments whereas voters of Labour party don't have much strong opinions on Europe Integration
- **political.knowledge:** Voters of both parties consider their political knowledge to be good

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).

Ans)

Encoding:

While observing the data we can see that columns having string values are **'vote'** and **'gender'**. Conventionally, the value in minority is labeled as 1 and other as 0 in classification problems. Since label encoder by default labels values in alphabetical order, so the labels for Conservative party (minority) and Labour party (majority) were manually labeled as 1 and 0 respectively. It can be referred from line [41] of notebook.

Gender column is nominal in nature hence it is one hot encoded using get_dummies and dropping the first column.

Scaling:

We can see among numerical columns except 'age', all are ordinal/ categorical in nature and are on a similar scale of around 1-5. 'Europe' column is on a different scale of 1-11. Normalization is applied on metric variables. There is no point in normalizing the categorical variables.

The proper treatment of ordinal features in machine learning is tricky. The two most common approaches are:

- Treat them as continuous: The main downside here is that this technique ignores the fact that the differences in levels may not be similar.

- Treat them as categorical: This option ignores the ordered nature of the variable. However, if the effect or importance of that ordering is not that big or all that important, we then can be sure that we would not be overstating any effects.

Keeping this in mind we will be treating them as continuous variables. Also we know for a fact that the differences in levels are similar for each assessment column so it is a plus point for our argument.

We can also observe that two columns have a different scale. 'Age' has a range from 24 to 93 and Europe has a scale of 1 – 11. Instead of bringing them on a similar scale using z-score and bringing their mean and standard deviation to 0 and 1 in order to process them as good data for modeling, binning was performed on them to bring them of similar scale of 1 – 5 in the similar ordinal fashion as follows:

Age		**Europe**	
Interval	**Label value**	**Interval**	**Label value**
24 – 38	1	1 – 2	1
39 – 53	2	3 – 4	2
54 – 68	3	5 – 6	3
69 – 83	4	7 – 8	4
84 – 98	5	9 – 10	5
		11	6

Table 4: Label value and intervals for binning

This method was performed in an attempt to bring columns on similar scales and range without affecting the distribution of data. They follow similar orderly fashion.

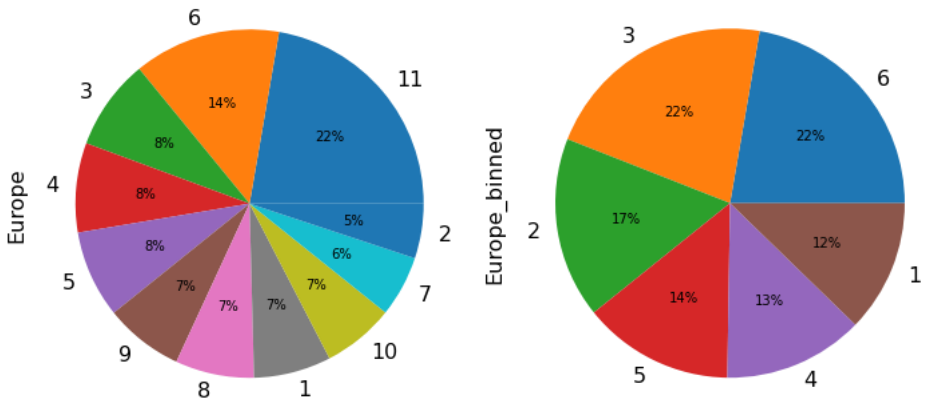


Figure 14: Comparison of Europe column before and after binning

In this way the difference in levels is maintained and also the distribution is uniform too.

The model performances were also checked without scaling the data too for comparison purpose on few models but it was found that binning was slightly better at not over-fitting the train sets for certain models like Decision Tree and Random forest.

The results were pretty much same for these approaches. But to bring all columns to similar scale so that normalization is not performed on the columns which actually change the scale of normalized column from other columns, and the approach to make sense so that same data can be used for all models for better performance and comparison of metrics we proceeded with the binning approach for this case study.

Data was split in the training and testing set in the ratio of 70:30 and can be referred from line [44] of notebook.

1.4) Apply Logistic Regression and LDA (Linear Discriminant Analysis).

Ans)

NOTE: While building all models below, first the base model with no parameters passed was built with just random_state if applicable to maintain consistency in results. After that grid search along with cross validation was performed iteratively to get the best models using different hyperparameters (codes from line [46] of the notebook). The performances of both models was compared using 10 fold cross validation scores and also other metrics like precision, recall and F1-score along with accuracy were selected to choose the optimal model among them for each algorithm as the given dataset is imbalanced. We will be paying more attention to recall, precision and F1-score of class 1 (conservative party) as they are underrepresented, and the other class is bound to perform better due to majority.

Cross validation is an approach to split the dataset into training and testing to make sure each observation becomes part of training and testing dataset to ensure the complete representation of data for a better model.

LOGISTIC REGRESSION:

Base Model	**GridSearchCV Model**
Classification Report	
Classification Report of the training data: precision recall f1-score support 0 0.86 0.91 0.88 754 1 0.74 0.64 0.68 307 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 0.86 0.88 0.87 303 1 0.76 0.73 0.74 153 accuracy 0.83 456 macro avg 0.81 0.80 0.81 456 weighted avg 0.83 0.83 0.83 456	Classification Report of the training data: precision recall f1-score support 0 0.86 0.91 0.88 754 1 0.74 0.64 0.68 307 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 0.86 0.88 0.87 303 1 0.76 0.73 0.74 153 accuracy 0.83 456 macro avg 0.81 0.80 0.81 456 weighted avg 0.83 0.83 0.83 456
Cross Validated Score	
Train set: 0.8323 **Test set:** 0.8422	**Train set:** 0.8332 **Test set:** 0.8422

Table 5: Comparison of Logistic Regression base and tuned models

We can see that the base as well as tuned model gave similar results. The CV scores are also same for both the models. Tuning did not improve the performance any further. So to save computational time and additional resources we would be going ahead with the base model.

LINEAR DISCRIMINANT ANALYSIS (LDA):

Base Model	GridSearchCV Model
Classification Report	
Classification Report of the training data: precision recall f1-score support 0 0.86 0.91 0.88 754 1 0.74 0.64 0.69 307 accuracy 0.83 1061 macro avg 0.80 0.78 0.79 1061 weighted avg 0.83 0.83 0.83 1061 Classification Report of the test data: precision recall f1-score support 0 0.87 0.88 0.88 303 1 0.76 0.73 0.75 153 accuracy 0.83 456 macro avg 0.81 0.81 0.81 456 weighted avg 0.83 0.83 0.83 456	Classification Report of the training data: precision recall f1-score support 0 0.87 0.91 0.89 754 1 0.74 0.65 0.69 307 accuracy 0.83 1061 macro avg 0.80 0.78 0.79 1061 weighted avg 0.83 0.83 0.83 1061 Classification Report of the test data: precision recall f1-score support 0 0.87 0.87 0.87 303 1 0.75 0.73 0.74 153 accuracy 0.83 456 macro avg 0.81 0.80 0.80 456 weighted avg 0.83 0.83 0.83 456
Cross Validated Score	
Train set: 0.8276 **Test set:** 0.8399	**Train set:** 0.8304 **Test set:** 0.8377

Table 6: Comparison of LDA base and tuned models

We can see that the base model performed slightly better in terms of recall of '1' after tuning but no change in performance was observed for test set. The CV scores are almost same for both the models. Tuning did not improve the performance any further. So to save computational time and additional resources we would be going ahead with the base model.

1.5) Apply KNN Model and Naïve Bayes Model. Interpret.

Ans)

KNN:

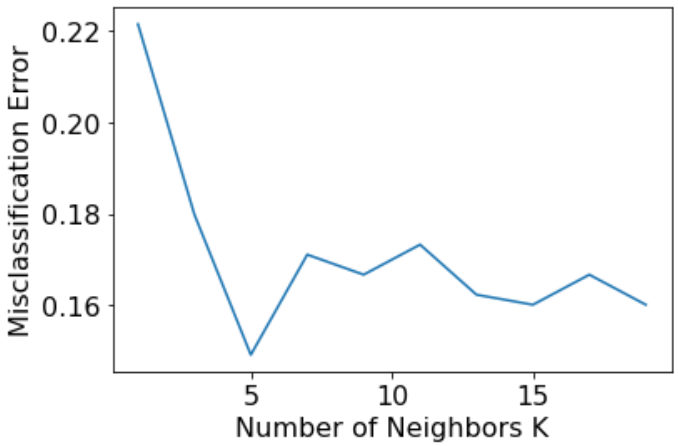


Figure 15: MCE vs NN

We found by checking the misclassification error on different nearest neighbor values that the least error was obtained for k = 5 which is also the default value for the KNN classifier.

Base Model	GridSearchCV Model
Classification Report	
Classification Report of the training data: precision recall f1-score support 0 0.90 0.91 0.91 754 1 0.78 0.75 0.76 307 accuracy 0.86 1061 macro avg 0.84 0.83 0.83 1061 weighted avg 0.86 0.86 0.86 1061 Classification Report of the test data: precision recall f1-score support 0 0.87 0.91 0.89 303 1 0.81 0.73 0.77 153 accuracy 0.85 456 macro avg 0.84 0.82 0.83 456 weighted avg 0.85 0.85 0.85 456	Classification Report of the training data: precision recall f1-score support 0 0.90 0.91 0.91 754 1 0.78 0.75 0.76 307 accuracy 0.86 1061 macro avg 0.84 0.83 0.83 1061 weighted avg 0.86 0.86 0.86 1061 Classification Report of the test data: precision recall f1-score support 0 0.87 0.91 0.89 303 1 0.81 0.73 0.77 153 accuracy 0.85 456 macro avg 0.84 0.82 0.83 456 weighted avg 0.85 0.85 0.85 456
Cross Validated Score	
Train set: 0.8181 **Test set:** 0.8067	**Train set:** 0.8304 **Test set:** 0.8377

Table 7: Comparison of KNN base and tuned models

We can see that the base as well as tuned model gave similar results. But the CV scores better for the tuned model. The computational time is also not much for tuning (checked using %%time in notebook). So in this case we will be going ahead with the tuned model.

NAÏVE BAYES:

Base Model	**GridSearchCV Model**
Classification Report	
Classification Report of the training data: precision recall f1-score support 0 0.88 0.90 0.89 754 1 0.74 0.68 0.71 307 accuracy 0.84 1061 macro avg 0.81 0.79 0.80 1061 weighted avg 0.84 0.84 0.84 1061 Classification Report of the test data: precision recall f1-score support 0 0.87 0.86 0.87 303 1 0.73 0.75 0.74 153 accuracy 0.82 456 macro avg 0.80 0.80 0.80 456 weighted avg 0.82 0.82 0.82 456	Classification Report of the training data: precision recall f1-score support 0 0.86 0.91 0.88 754 1 0.74 0.64 0.68 307 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 0.86 0.88 0.87 303 1 0.76 0.73 0.74 153 accuracy 0.83 456 macro avg 0.81 0.80 0.81 456 weighted avg 0.83 0.83 0.83 456
Cross Validated Score	
Train set: 0.8332 **Test set:** 0.8224	**Train set:** 0.8332 **Test set:** 0.8422

Table 8: Comparison of Naïve Bayes base and tuned models

Naïve Bayes as such doesn't have many hyperparameters as such. But trying to tune the model decreased its performance in terms of precision and recall although the accuracy is higher. But the other metrics matter more hence we will be going ahead with the base model for this case.

1.6) Model Tuning, Bagging and Boosting.

Ans)

Decision Tree:

Base Model	GridSearchCV Model
Classification Report	
Classification Report of the training data: precision recall f1-score support 0 0.98 1.00 0.99 754 1 1.00 0.95 0.97 307 accuracy 0.99 1061 macro avg 0.99 0.98 0.98 1061 weighted avg 0.99 0.99 0.99 1061 Classification Report of the test data: precision recall f1-score support 0 0.83 0.86 0.85 303 1 0.70 0.65 0.68 153 accuracy 0.79 456 macro avg 0.77 0.76 0.76 456 weighted avg 0.79 0.79 0.79 456	Classification Report of the training data: precision recall f1-score support 0 0.88 0.90 0.89 754 1 0.74 0.71 0.72 307 accuracy 0.84 1061 macro avg 0.81 0.80 0.81 1061 weighted avg 0.84 0.84 0.84 1061 Classification Report of the test data: precision recall f1-score support 0 0.85 0.88 0.87 303 1 0.75 0.69 0.72 153 accuracy 0.82 456 macro avg 0.80 0.79 0.79 456 weighted avg 0.82 0.82 0.82 456
Cross Validated Score	
Train set: 0.7719 **Test set:** 0.7238	**Train set:** 0.8237 **Test set:** 0.7716

Table 9: Comparison of Decision Tree base and tuned models

The base model is a classic case of overfitting where data performed really well on train set but poorly on test set as can be seen from the performance metrics above. Tuning helped the overfitting problem as well as improved the test set performance hence tuned model will be considered here. This model will further be used for performing bagging.

BAGGING:

Base Model	GridSearchCV Model
Classification Report	
Classification Report of the training data: precision recall f1-score support 0 0.87 0.92 0.90 754 1 0.78 0.67 0.72 307 accuracy 0.85 1061 macro avg 0.82 0.80 0.81 1061 weighted avg 0.85 0.85 0.85 1061 Classification Report of the test data: precision recall f1-score support 0 0.83 0.91 0.87 303 1 0.78 0.63 0.70 153 accuracy 0.82 456 macro avg 0.80 0.77 0.78 456 weighted avg 0.81 0.82 0.81 456	Classification Report of the training data: precision recall f1-score support 0 0.88 0.94 0.91 754 1 0.82 0.67 0.74 307 accuracy 0.86 1061 macro avg 0.85 0.81 0.82 1061 weighted avg 0.86 0.86 0.86 1061 Classification Report of the test data: precision recall f1-score support 0 0.84 0.92 0.88 303 1 0.81 0.66 0.73 153 accuracy 0.83 456 macro avg 0.83 0.79 0.80 456 weighted avg 0.83 0.83 0.83 456
Cross Validated Score	
Train set: 0.8209 **Test set:** 0.8026	**Train set:** 0.8341 **Test set:** 0.8200

Table 10: Comparison of Bagging base and tuned models

Tuning has certainly helped in improving the train and test set scores but the computation time is also slightly high as compared to previous models. We will be going ahead with tuned model still as it has improved the performance w.r.t. all aspects in consideration.

RANDOM FOREST:

Base Model	GridSearchCV Model
Classification Report	
Classification Report of the training data: precision recall f1-score support 0 0.99 0.99 0.99 754 1 0.98 0.97 0.98 307 accuracy 0.99 1061 macro avg 0.98 0.98 0.98 1061 weighted avg 0.99 0.99 0.99 1061 Classification Report of the test data: precision recall f1-score support 0 0.86 0.91 0.89 303 1 0.81 0.71 0.76 153 accuracy 0.85 456 macro avg 0.84 0.81 0.82 456 weighted avg 0.84 0.85 0.84 456	Classification Report of the training data: precision recall f1-score support 0 0.91 0.94 0.93 754 1 0.84 0.78 0.81 307 accuracy 0.89 1061 macro avg 0.88 0.86 0.87 1061 weighted avg 0.89 0.89 0.89 1061 Classification Report of the test data: precision recall f1-score support 0 0.86 0.91 0.88 303 1 0.79 0.71 0.75 153 accuracy 0.84 456 macro avg 0.83 0.81 0.82 456 weighted avg 0.84 0.84 0.84 456
Cross Validated Score	
Train set: 0.8087 **Test set:** 0.7872	**Train set:** 0.8369 **Test set:** 0.8004

Table 11: Comparison of Random Forest base and tuned models

Random forest is an ensemble technique which internally uses bagging and bootstrap aggregation mechanism to create decision trees and model building.

The base model here too is a classic case of overfitting where data performed really well on train set but poorly on test set as can be seen from the performance metrics above. Tuning helped the overfitting problem as well as improved the test set performance hence tuned model will be considered here.

Boosting:

1. **Adaptive Boosting:**

Base Model	GridSearchCV Model
Classification Report	
Classification Report of the training data: precision recall f1-score support 0 0.87 0.91 0.89 754 1 0.74 0.67 0.71 307 accuracy 0.84 1061 macro avg 0.81 0.79 0.80 1061 weighted avg 0.84 0.84 0.84 1061 Classification Report of the test data: precision recall f1-score support 0 0.84 0.88 0.86 303 1 0.74 0.68 0.71 153 accuracy 0.81 456 macro avg 0.79 0.78 0.79 456 weighted avg 0.81 0.81 0.81 456	Classification Report of the training data: precision recall f1-score support 0 0.88 0.73 0.80 754 1 0.53 0.75 0.62 307 accuracy 0.74 1061 macro avg 0.70 0.74 0.71 1061 weighted avg 0.78 0.74 0.75 1061 Classification Report of the test data: precision recall f1-score support 0 0.86 0.73 0.79 303 1 0.59 0.76 0.66 153 accuracy 0.74 456 macro avg 0.72 0.74 0.72 456 weighted avg 0.77 0.74 0.75 456
Cross Validated Score	
Train set: 0.8351 **Test set:** 0.7831	**Train set:** 0.7352 **Test set:** 0.7284

Table 12: Comparison of Adaptive Boosting base and tuned models

Adaboost helps you combine multiple “weak classifiers” into a single “strong classifier”. Decision stumps are the weak classifiers used by default. After tuning we could find a good boost in the performance metrics in terms of recall, but the precision was pretty low which also brought down the F1-score and also reduced the accuracy. So in this case to save computational time, resources and have good performance we will be proceeding with the base model.

2. Gradient Boosting:

Base Model

Classification Report of the training data:

	precision	recall	f1-score	support
0	0.90	0.93	0.92	754
1	0.82	0.75	0.78	307
accuracy			0.88	1061
macro avg	0.86	0.84	0.85	1061
weighted avg	0.88	0.88	0.88	1061

Classification Report of the test data:

	precision	recall	f1-score	support
0	0.86	0.91	0.88	303
1	0.79	0.71	0.75	153
accuracy			0.84	456
macro avg	0.83	0.81	0.82	456
weighted avg	0.84	0.84	0.84	456

Cross Validated Score

Train set: 0.8369
Test set: 0.7938

GridSearchCV Model

Classification Report of the training data:

	precision	recall	f1-score	support
0	0.89	0.94	0.91	754
1	0.82	0.72	0.77	307
accuracy			0.87	1061
macro avg	0.86	0.83	0.84	1061
weighted avg	0.87	0.87	0.87	1061

Classification Report of the test data:

	precision	recall	f1-score	support
0	0.86	0.90	0.88	303
1	0.78	0.71	0.75	153
accuracy			0.84	456
macro avg	0.82	0.81	0.81	456
weighted avg	0.84	0.84	0.84	456

Cross Validated Score

Train set: 0.8379
Test set: 0.8071

Table 13: Comparison of Gradient Boosting base and tuned models

We can see that the base as well as tuned model gave similar results. The CV scores are also same for both the models. Tuning did not improve the performance any further. It Infact decreased the performance of train set slightly. So to save computational time and additional resources we would be going ahead with the base model.

3. Extreme Gradient Boosting:

Base Model	GridSearchCV Model
Classification Report	
Classification Report of the training data: precision recall f1-score support 0 0.97 0.98 0.98 754 1 0.96 0.92 0.94 307 accuracy 0.97 1061 macro avg 0.96 0.95 0.96 1061 weighted avg 0.97 0.97 0.97 1061 Classification Report of the test data: precision recall f1-score support 0 0.85 0.89 0.87 303 1 0.77 0.69 0.73 153 accuracy 0.83 456 macro avg 0.81 0.79 0.80 456 weighted avg 0.82 0.83 0.82 456	Classification Report of the training data: precision recall f1-score support 0 0.97 0.98 0.98 754 1 0.96 0.92 0.94 307 accuracy 0.97 1061 macro avg 0.96 0.95 0.96 1061 weighted avg 0.97 0.97 0.97 1061 Classification Report of the test data: precision recall f1-score support 0 0.85 0.89 0.87 303 1 0.77 0.69 0.73 153 accuracy 0.83 456 macro avg 0.81 0.79 0.80 456 weighted avg 0.82 0.83 0.82 456
Cross Validated Score	
Train set: 0.8011 **Test set:** 0.7829	**Train set:** 0.8350 **Test set:** 0.8026

Table 14: Comparison of Extreme Gradient Boosting base and tuned models

Even after trying much iteration of different combinations using gridsearch, the training set was still overfitting and performance didn't improve.

So randomsearch with CV was used to search for values over a wider range and an optimal model was obtained with following metrics:

Classification Report of the training data:

	precision	recall	f1-score	support
0	0.88	0.91	0.90	754
1	0.76	0.69	0.73	307
accuracy			0.85	1061
macro avg	0.82	0.80	0.81	1061
weighted avg	0.85	0.85	0.85	1061

Classification Report of the test data:

	precision	recall	f1-score	support
0	0.86	0.88	0.87	303
1	0.74	0.71	0.72	153
accuracy			0.82	456
macro avg	0.80	0.79	0.80	456
weighted avg	0.82	0.82	0.82	456

Figure 16: Classification Report for RandomSearchCV model

CV Scores: Train set: 0.8247; **Test set:** 0.8093

Since the computation time is less and performance has also improved, we will be using model formed using the random search.

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

Ans)

<u>Logistic Regression:</u>

Confusion Matrix:

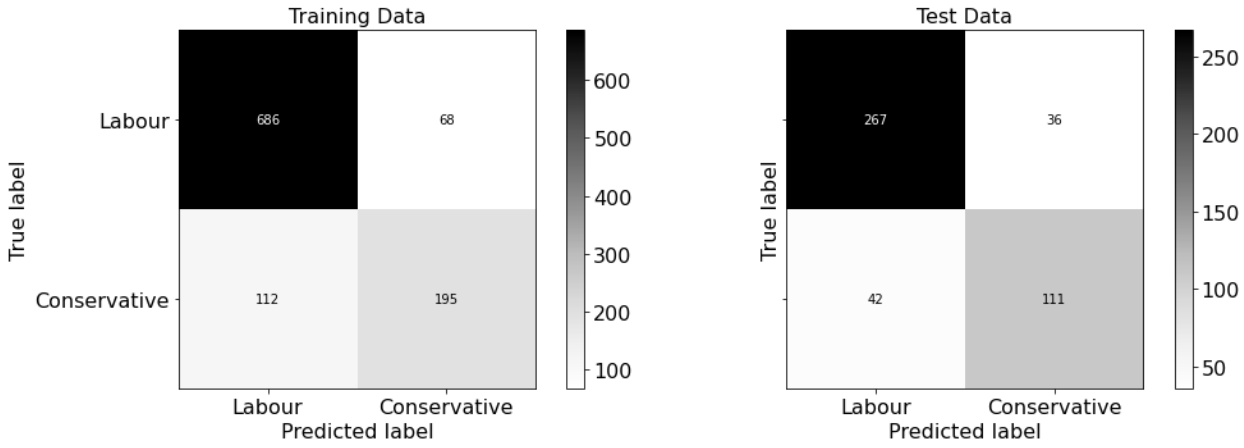


Figure 17: Confusion matrix of Logistic Regression

ROC Curve:

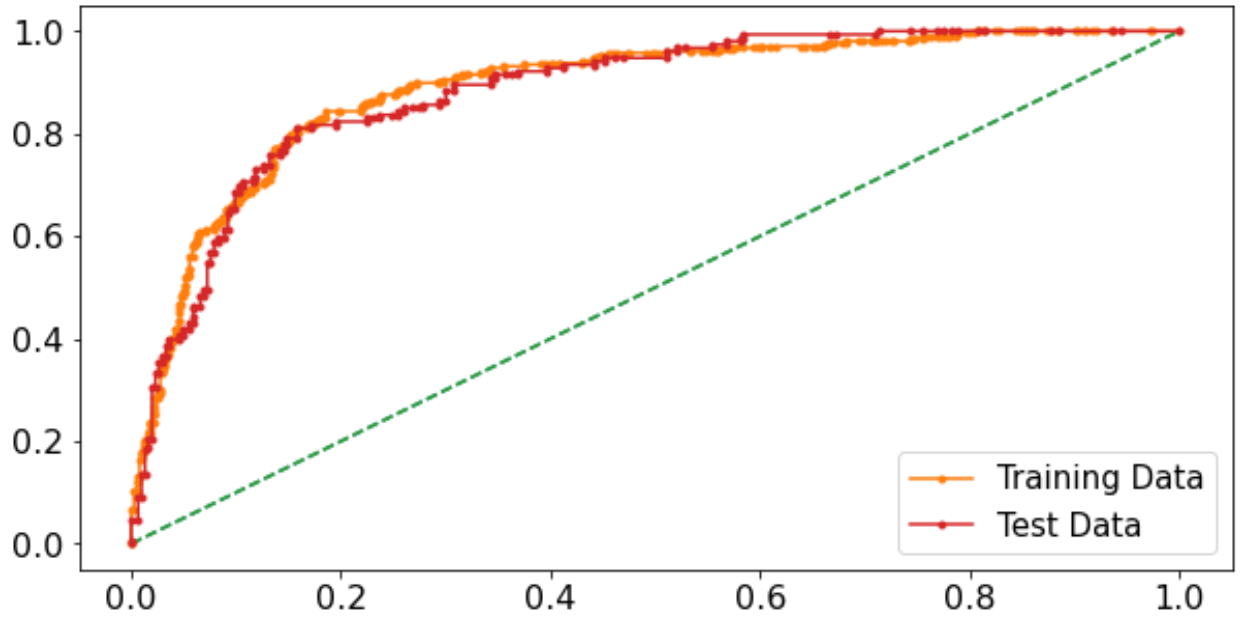


Figure 18: ROC curve of Logistic Regression

LDA:

Confusion Matrix:

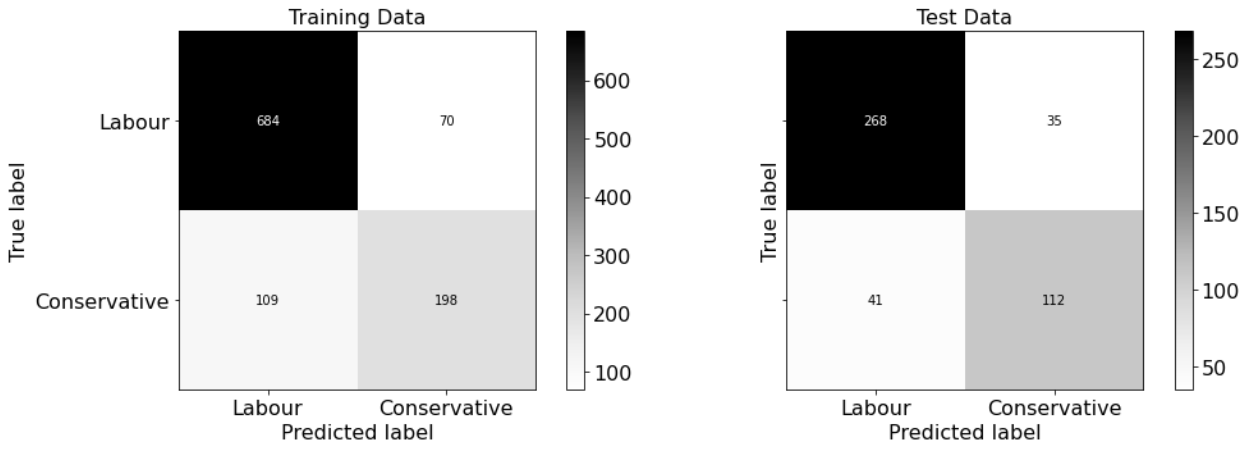


Figure 19: Confusion matrix of LDA

ROC Curve:

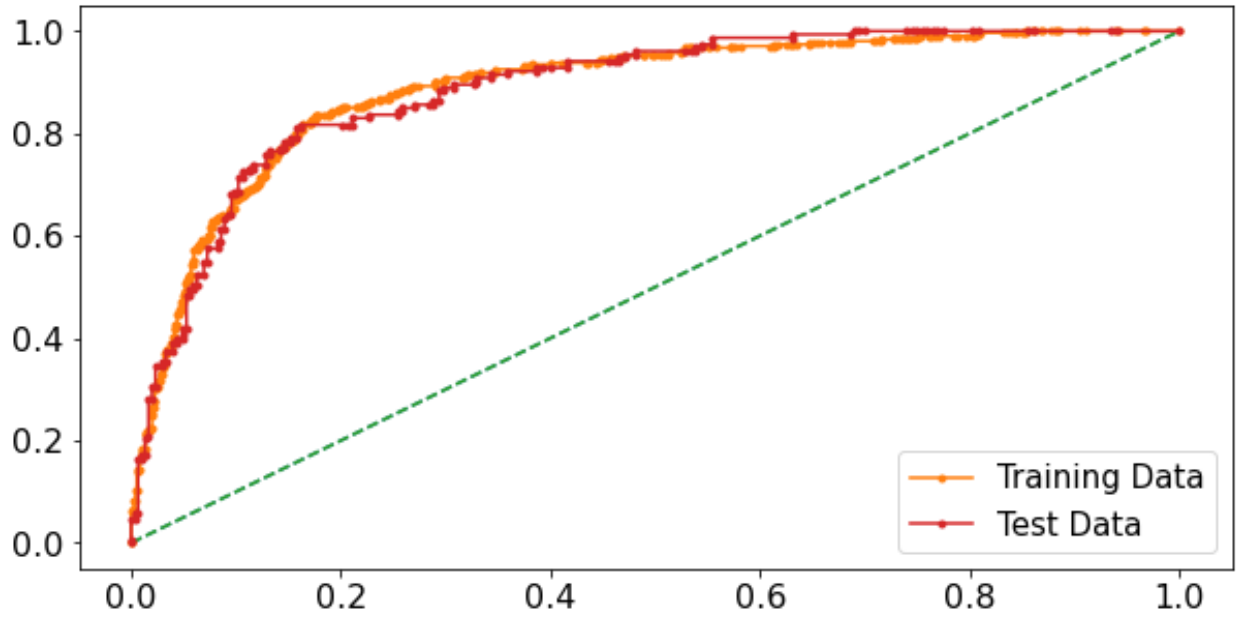


Figure 20: ROC curve of LDA

KNN:

Confusion Matrix:

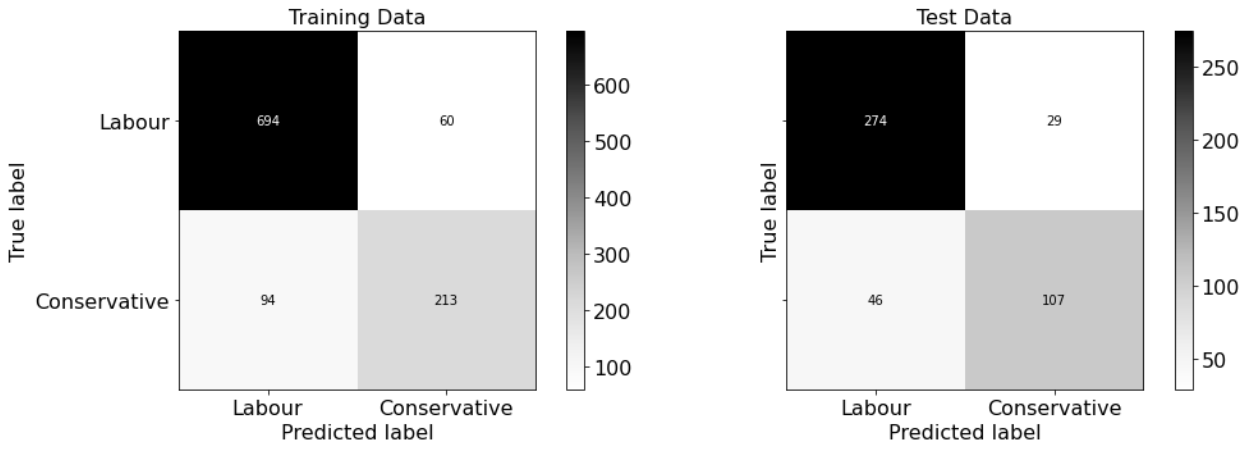


Figure 21: Confusion matrix of KNN

ROC Curve:

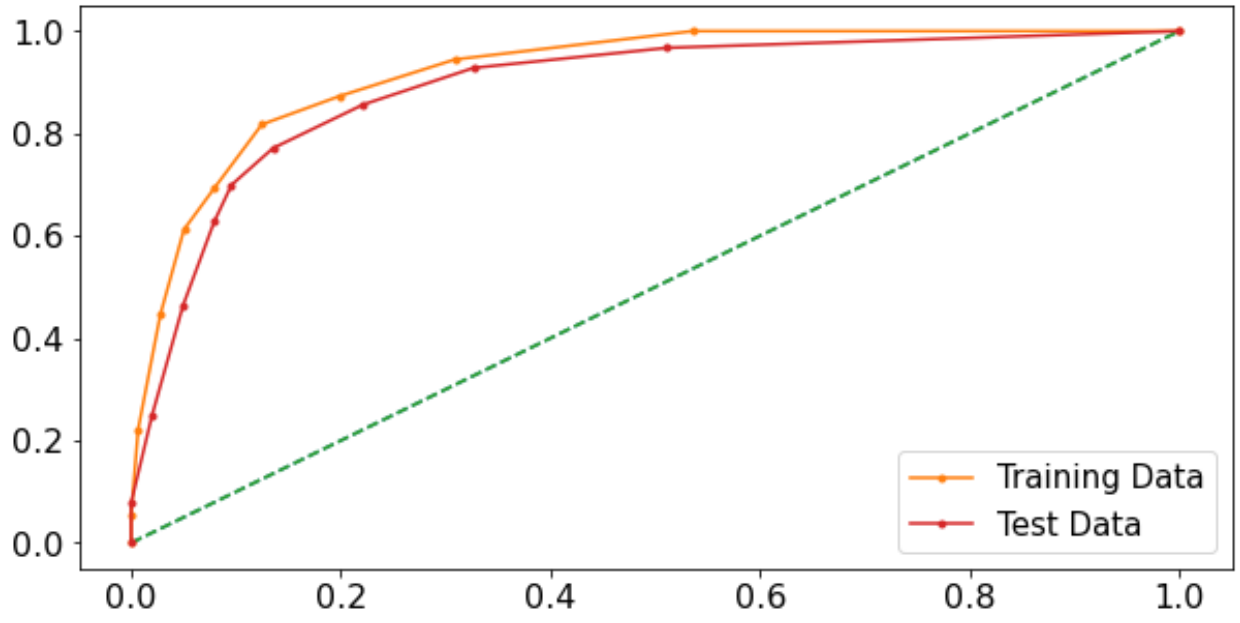


Figure 22: ROC curve of KNN

Naïve Bayes:

Confusion Matrix:

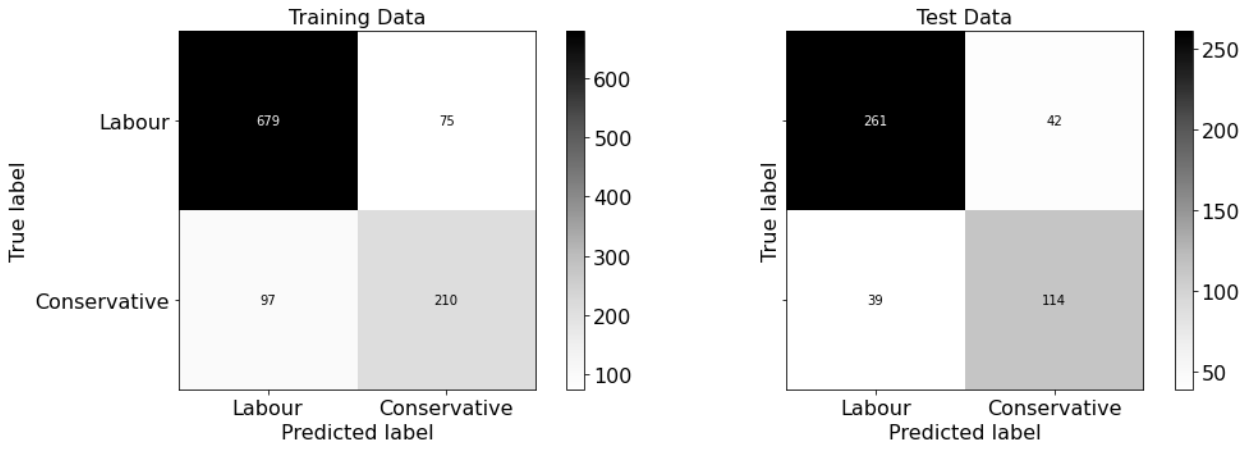


Figure 23: Confusion matrix of Naïve Bayes

ROC Curve:

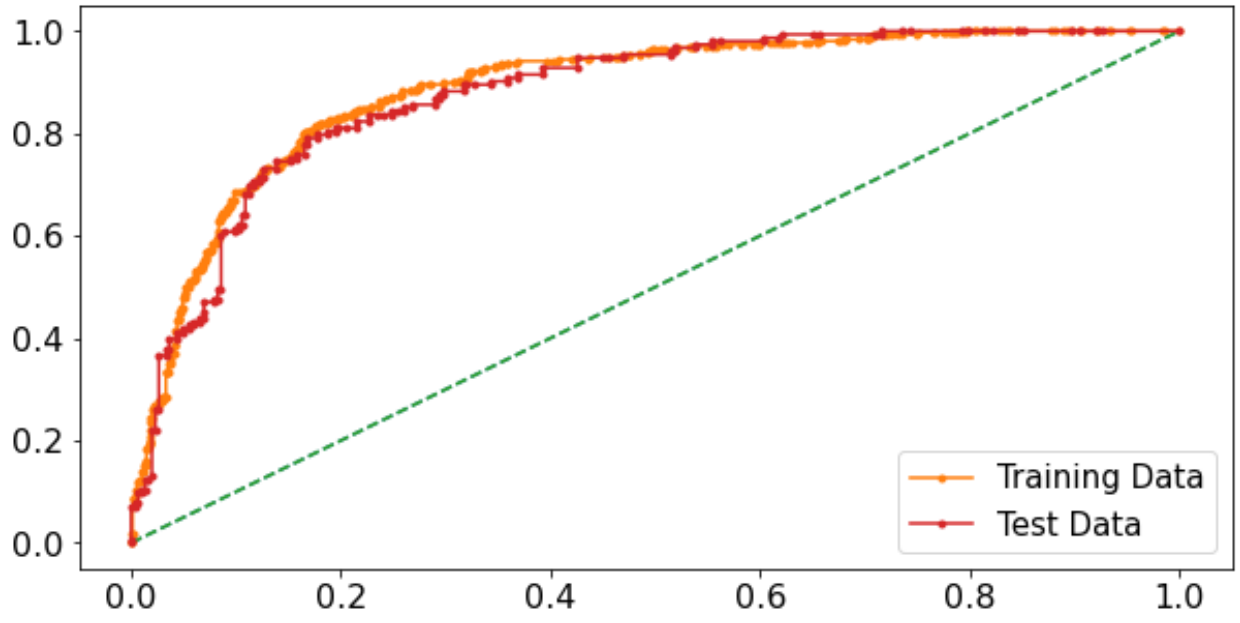


Figure 24: ROC curve of Naïve Bayes

Decision Tree:

Confusion Matrix:

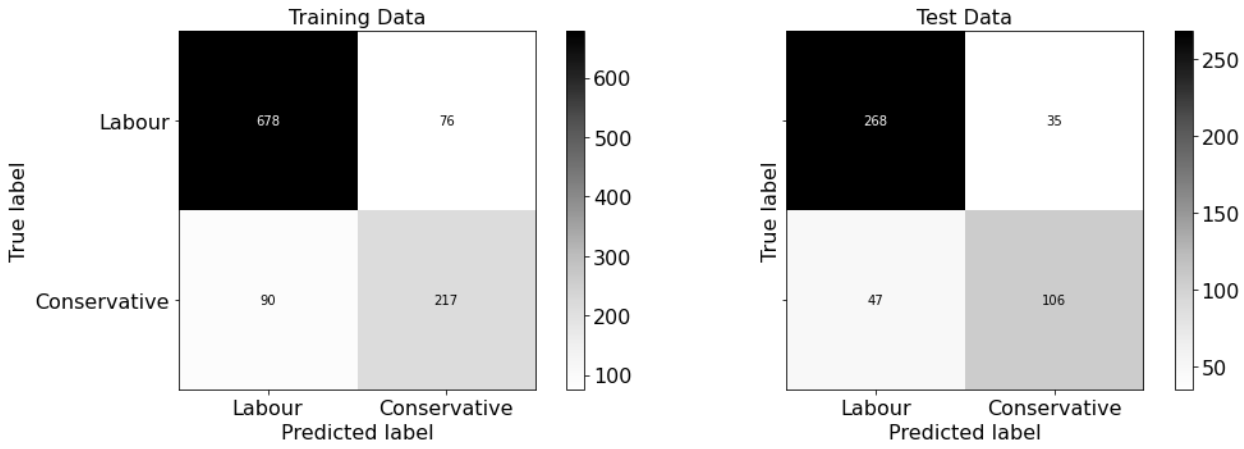


Figure 25: Confusion matrix of Decision Tree

ROC Curve:

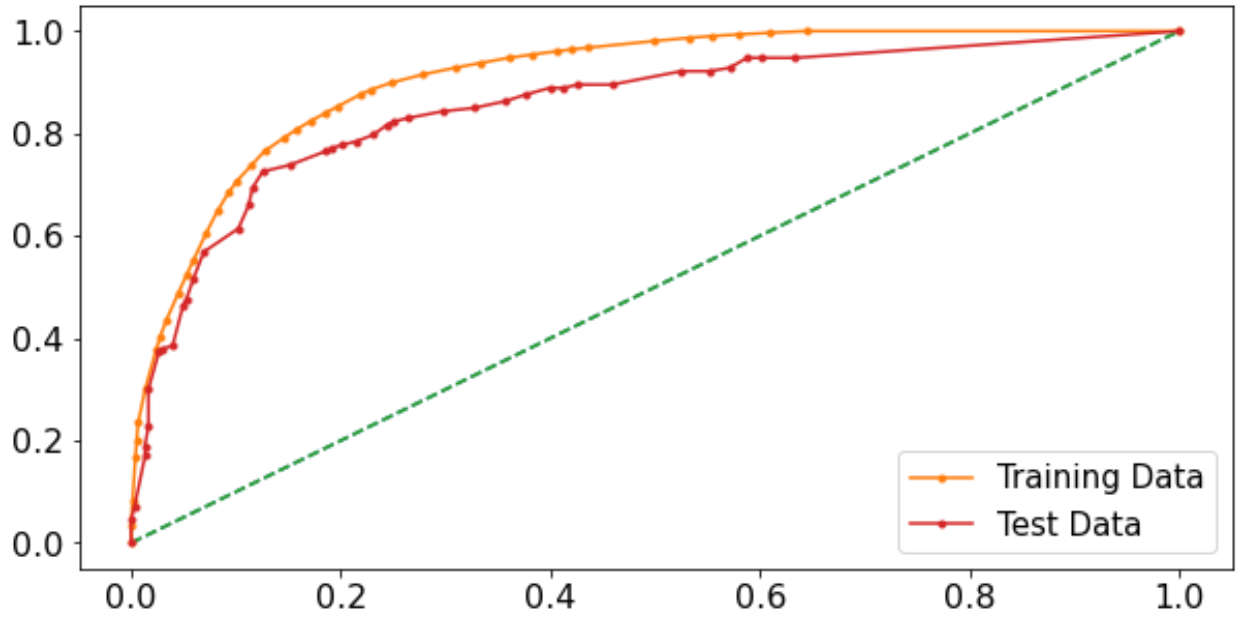


Figure 26: ROC curve of Decision Tree

Bagging:

Confusion Matrix:

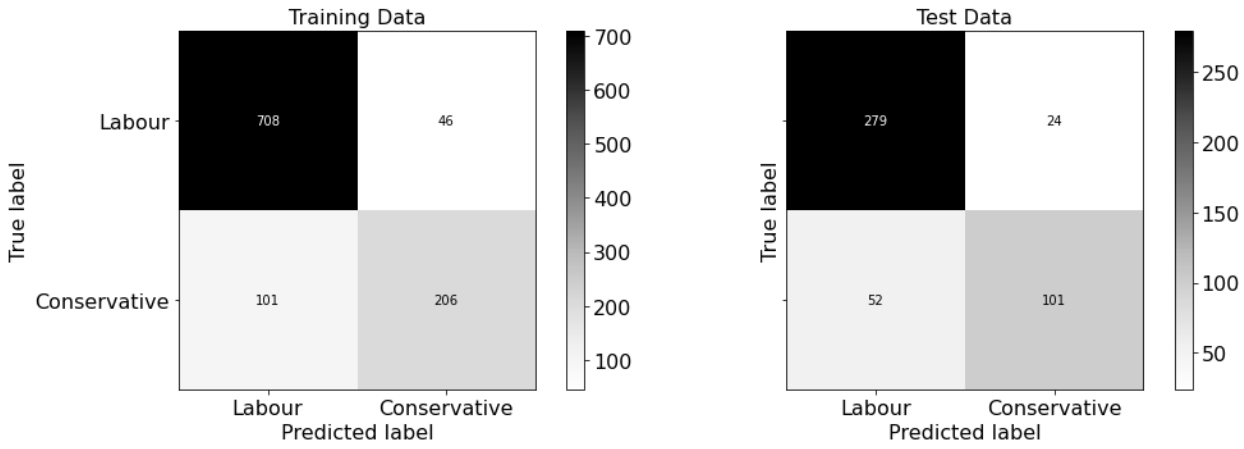


Figure 27: Confusion matrix of Bagging

ROC Curve:

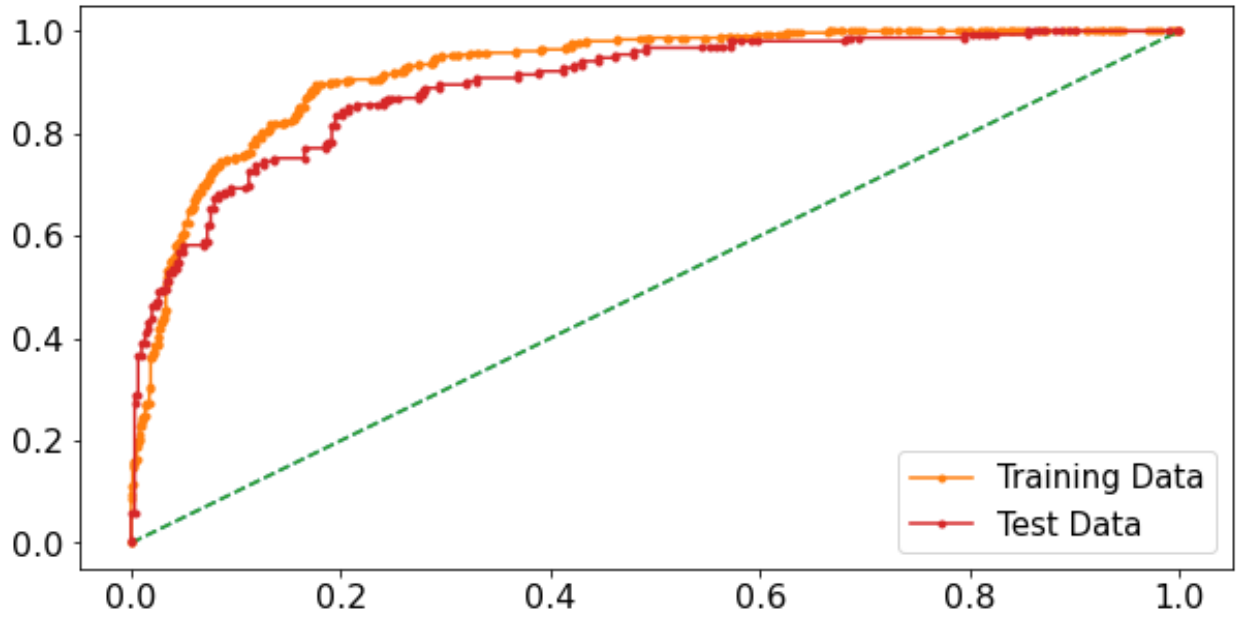


Figure 28: ROC curve of Bagging

Random Forest:

Confusion Matrix:

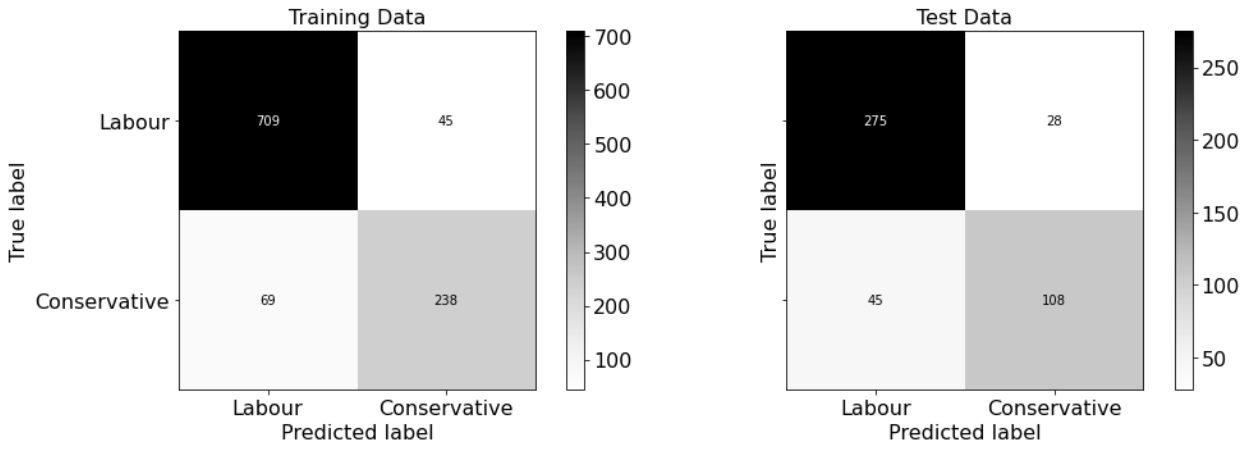


Figure 29: Confusion matrix of Random Forest

ROC Curve:

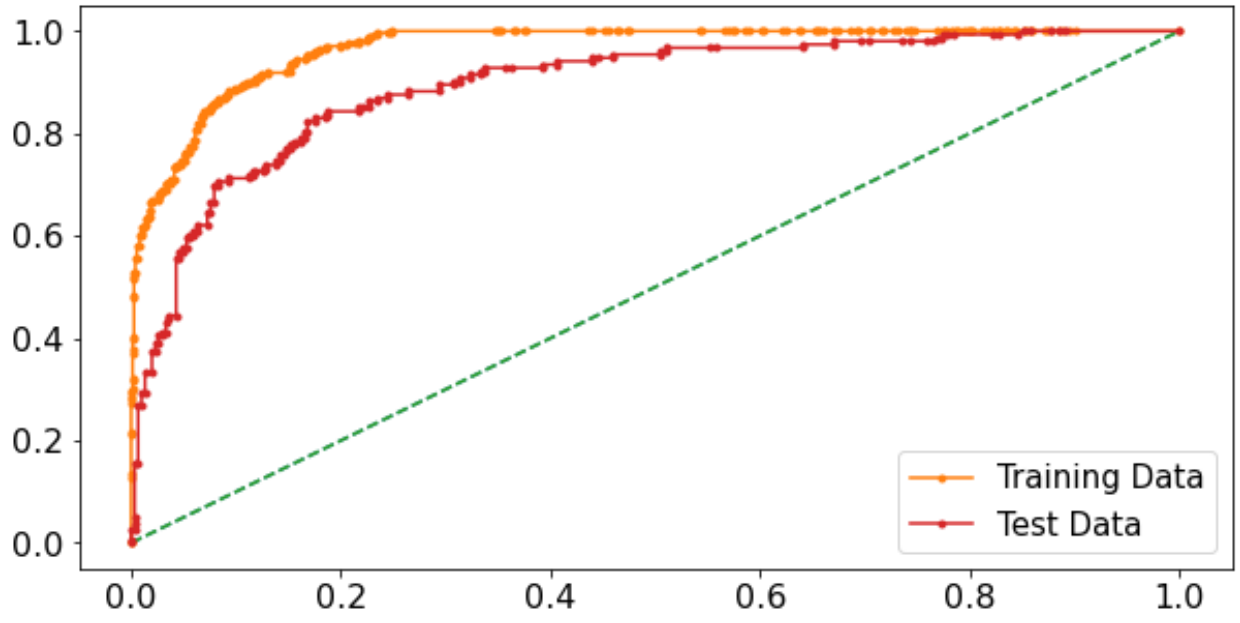


Figure 30: ROC curve of Random Forest

Boosting:

1. **Adaptive Boosting:**

Confusion Matrix:

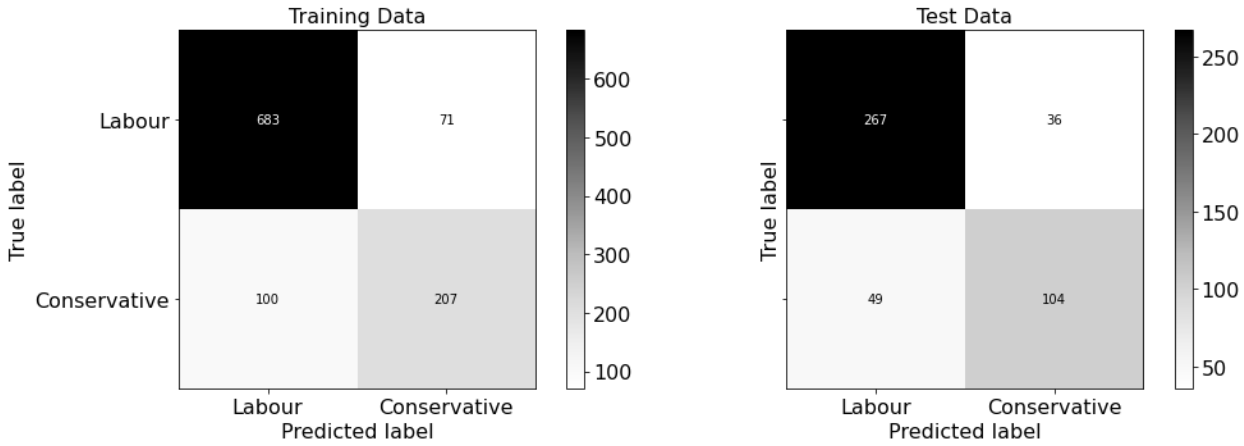


Figure 31: Confusion matrix of Adaptive Boosting

ROC Curve:

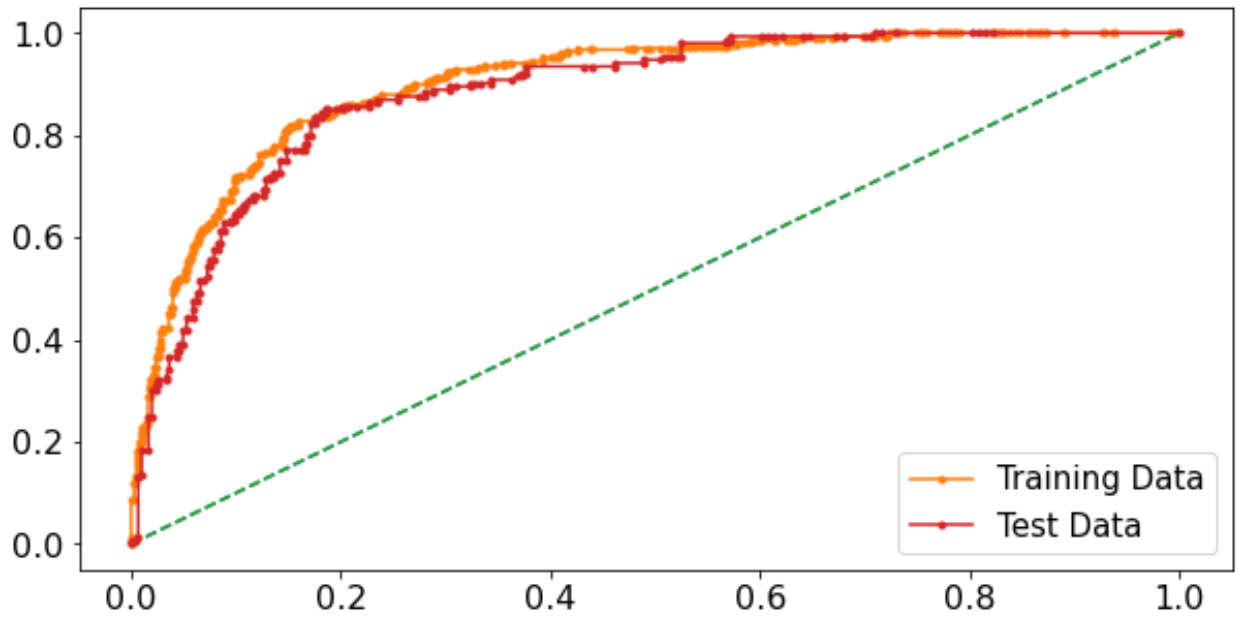


Figure 32: ROC curve of Adaptive Boosting

2. Gradient Boosting:

Confusion Matrix:

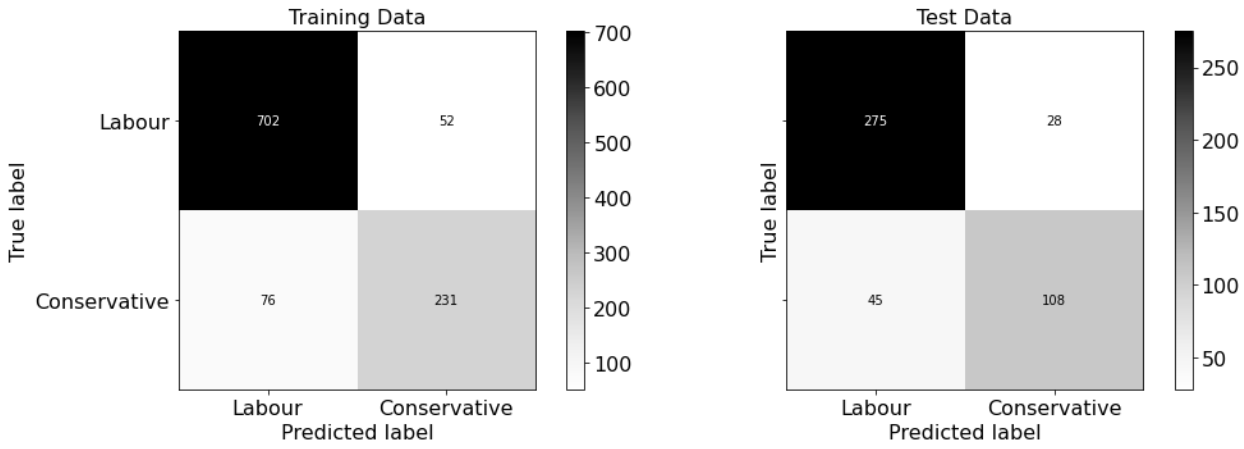


Figure 33: Confusion matrix of Gradient Boosting

ROC Curve:

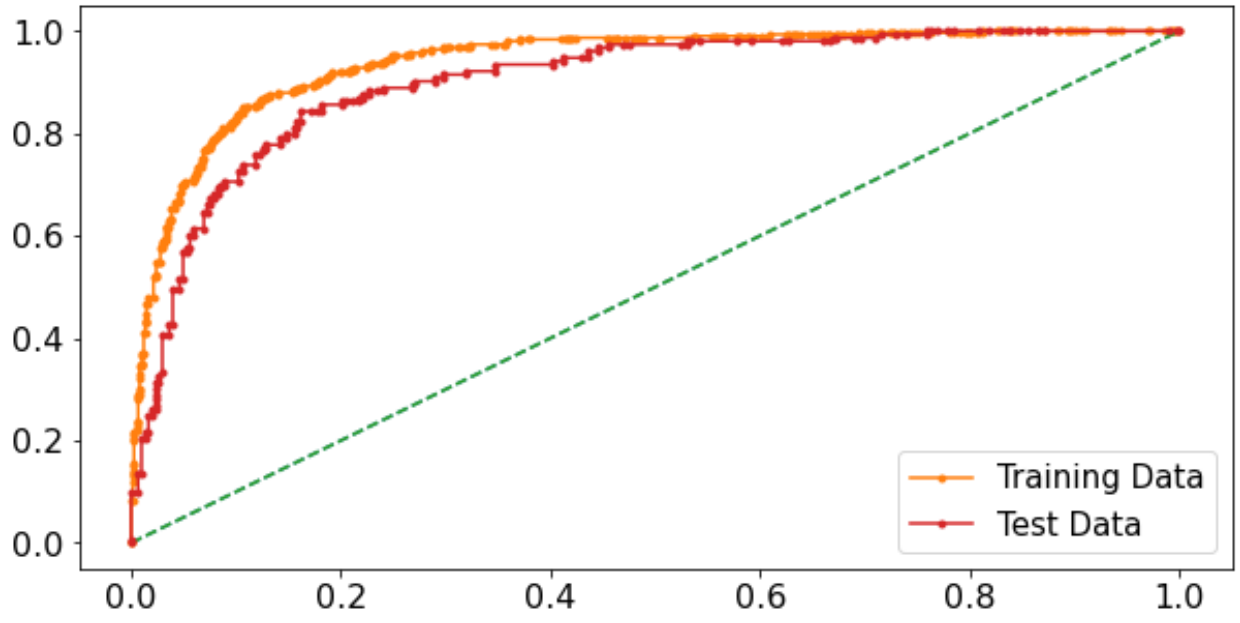


Figure 34: ROC curve of Gradient Boosting

3. Extreme Gradient Boosting:

Confusion Matrix:

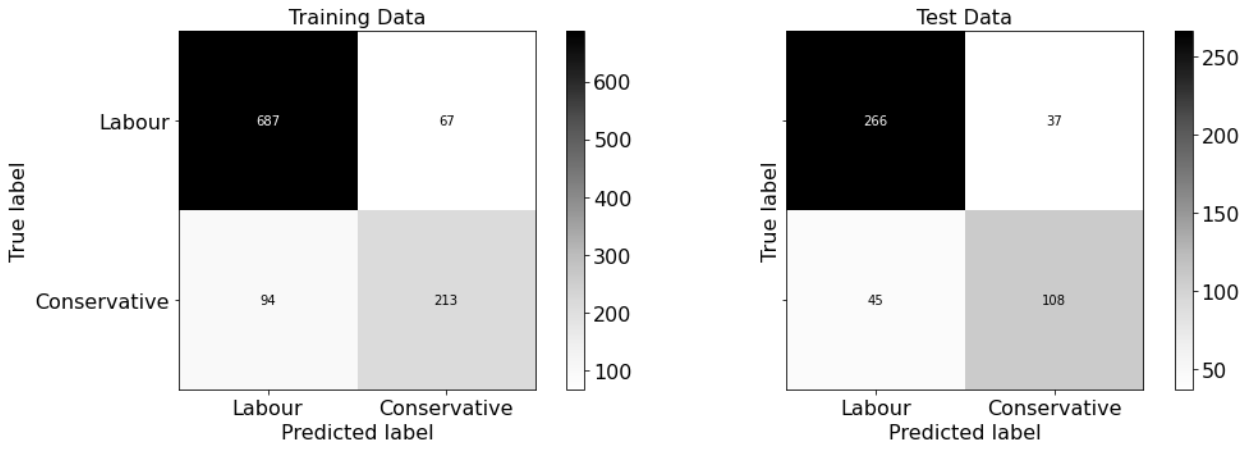


Figure 35: Confusion matrix of Extreme Gradient Boosting

ROC Curve:

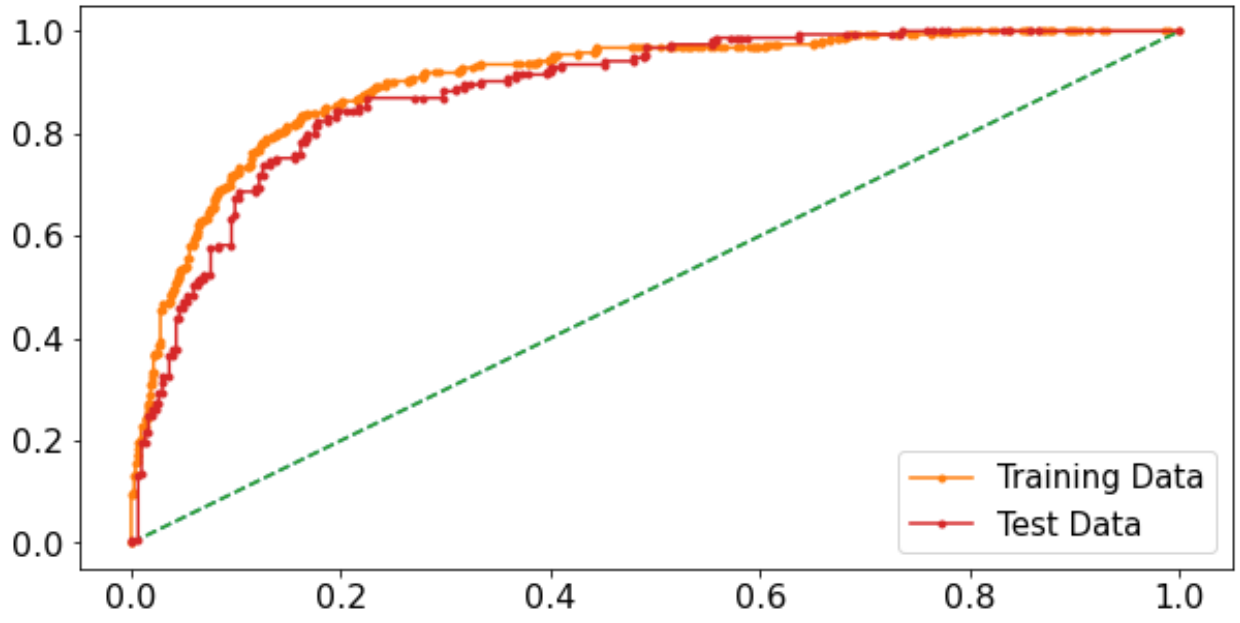


Figure 36: ROC curve of Extreme Gradient Boosting

By comparing the true positives (Actual Conservative party predicted as Conservative party) and true negatives (Actual Labour party predicted as Labour party) values we see that 3 models have performed really well as compared to others with are Bagging, Random Forest and Gradient Boosting with (TN,TP) values for train set as (708,206), (709,238) and (702,231) and test set as (279,101), (278,108) and (275,108) respectively.

Comparison of 'Accuracy', 'AUC', 'Recall','Precision','F1 Score' for all models

By comparing the important metrics like accuracy, AUC (area under the curve), recall, precision and F1-scores for all models we get the following results:

Note: The values were taken for the class 1 (conservative party) as it is the underrepresented class in this case and we need to get the best results for it as the results for other class is better in all aspects due to its majority.

	LR Train	LR Test	LDA Train	LDA Test	KNN Train	KNN Test	NB Train	NB Test	DT Train	DT Test	BAG Train	BAG Test	RF Train	RF Test	ADB Train	ADB Test	GB Train	GB Test	XGB Train	XGB Test
Accuracy	0.830	0.829	0.831	0.833	0.855	0.836	0.838	0.822	0.844	0.820	0.861	0.833	0.893	0.840	0.839	0.814	0.879	0.840	0.848	0.820
AUC	0.889	0.883	0.888	0.887	0.921	0.890	0.887	0.877	0.908	0.855	0.922	0.895	0.967	0.894	0.903	0.885	0.940	0.901	0.904	0.884
Recall	0.635	0.725	0.645	0.732	0.694	0.699	0.684	0.745	0.707	0.693	0.671	0.660	0.775	0.706	0.674	0.680	0.752	0.706	0.694	0.706
Precision	0.741	0.755	0.739	0.762	0.780	0.787	0.737	0.731	0.741	0.752	0.817	0.808	0.841	0.794	0.745	0.743	0.816	0.794	0.761	0.745
F1 Score	0.684	0.740	0.689	0.747	0.734	0.740	0.709	0.738	0.723	0.721	0.737	0.727	0.807	0.747	0.708	0.710	0.783	0.747	0.726	0.725

Table 15: DataFrame of important performance metrics of all models

Observing train and test set separately too,

	LR Train	LDA Train	KNN Train	NB Train	DT Train	BAG Train	RF Train	ADB Train	GB Train	XGB Train
Accuracy	0.830	0.831	0.855	0.838	0.844	0.861	0.893	0.839	0.879	0.848
AUC	0.889	0.888	0.921	0.887	0.908	0.922	0.967	0.903	0.940	0.904
Recall	0.635	0.645	0.694	0.684	0.707	0.671	0.775	0.674	0.752	0.694
Precision	0.741	0.739	0.780	0.737	0.741	0.817	0.841	0.745	0.816	0.761
F1 Score	0.684	0.689	0.734	0.709	0.723	0.737	0.807	0.708	0.783	0.726

Table 16: DataFrame of important performance metrics of train sets of all models

	LR Test	LDA Test	KNN Test	NB Test	DT Test	BAG Test	RF Test	ADB Test	GB Test	XGB Test
Accuracy	0.829	0.833	0.836	0.822	0.820	0.833	0.840	0.814	0.840	0.820
AUC	0.883	0.887	0.890	0.877	0.855	0.895	0.894	0.885	0.901	0.884
Recall	0.725	0.732	0.699	0.745	0.693	0.660	0.706	0.680	0.706	0.706
Precision	0.755	0.762	0.787	0.731	0.752	0.808	0.794	0.743	0.794	0.745
F1 Score	0.740	0.747	0.740	0.738	0.721	0.727	0.747	0.710	0.747	0.725

Table 15: DataFrame of important performance metrics of test sets of all models

Top 4-5 highest valued models for each metric: (In descending order)

Train:

- **Accuracy**: Random Forest, Gradient, KNN
- **AUC:** Random Forest, Gradient Boosting, Bagging, KNN
- **Recall:** Random Forest, Gradient Boosting, Decision Tree, Extreme Gradient Boosting, KNN
- **Precision:** Random Forest, Bagging, Gradient Boosting, KNN

- **F1-scores:** Random Forest, Gradient Boosting, KNN,

Test:

- **Accuracy**: Random Forest, Gradient, KNN, Bagging, LDA
- **AUC:** Gradient Boosting, Bagging, Random Forest, KNN
- **Recall:** Naive Bayes, LDA, Logistic Regression, Gradient Boosting, Extreme Gradient Boosting,
- **Precision:** Bagging, Random Forest, Gradient Boosting, KNN
- **F1-scores:** Random Forest, Gradient Boosting, LDA

Comparing the ROC curve of all models for both training and testing data:

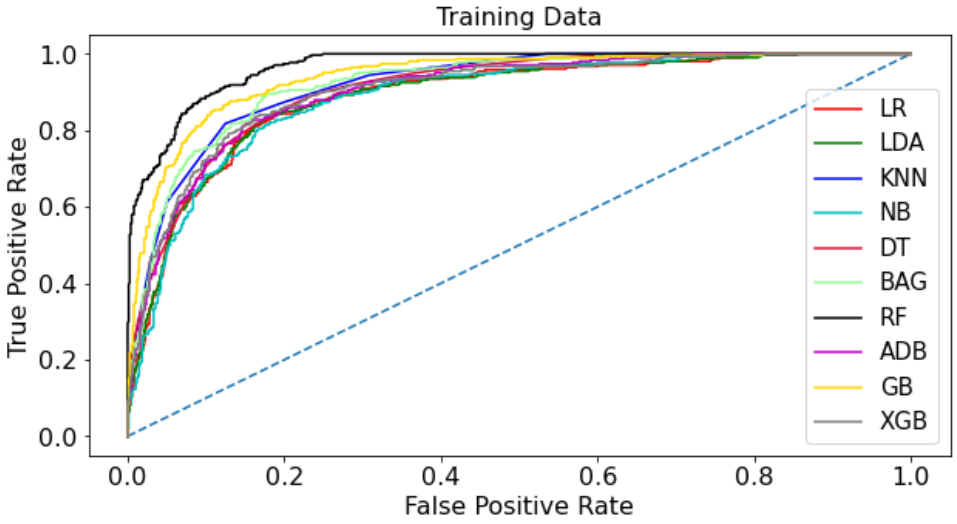


Figure 37: ROC curve comparison for training data

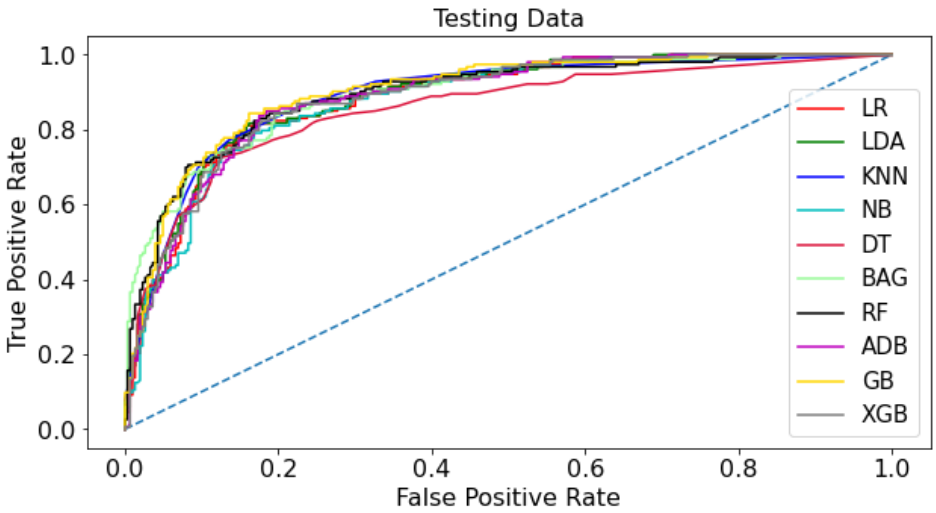


Figure 38: ROC curve comparison for testing data

We can see by observing both ROC curves that Random Forest and Gradient boosting models has performed very well than other models and they also have the highest AUC. Not only that they have highest values for all the important metrics considered for this problem in both train and test set.

The model which is has performed the best out of all the models considering all the important metrics is the Random Forest model followed very closely by the Gradient Boosting model and the Bagging model.

1.8) Based on these predictions, what are the insights?

Ans)

By checking the feature importance of all the features using three different algorithms, we get the following results:

Decision Tree:

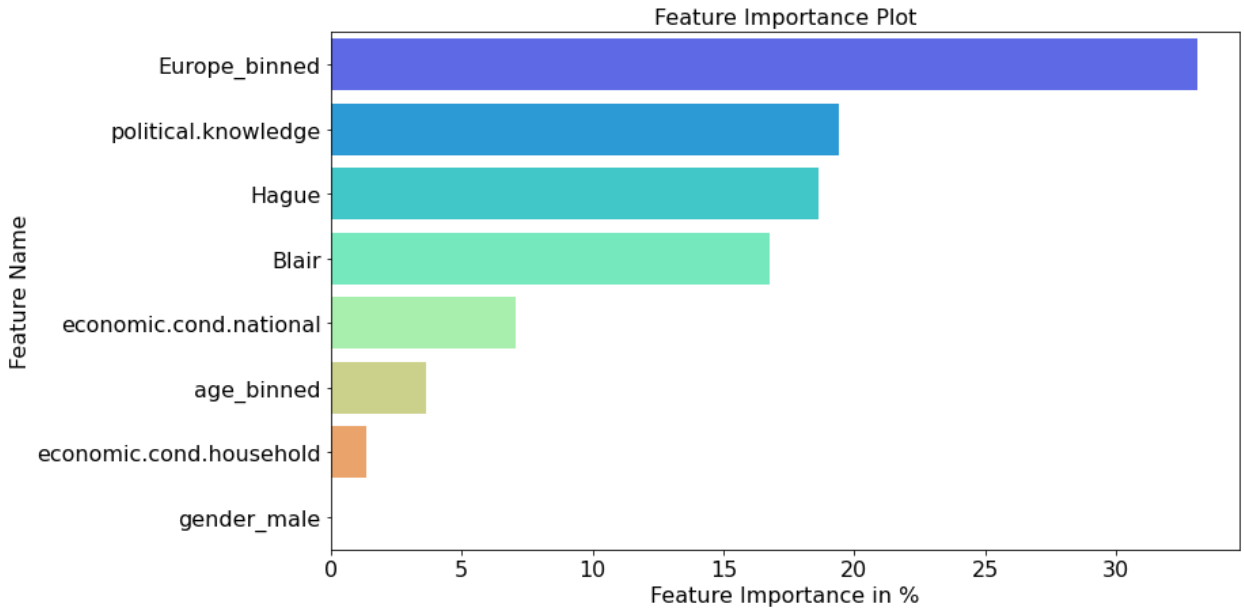


Figure 39: Feature importance as per Decision Tree

Random Forest:

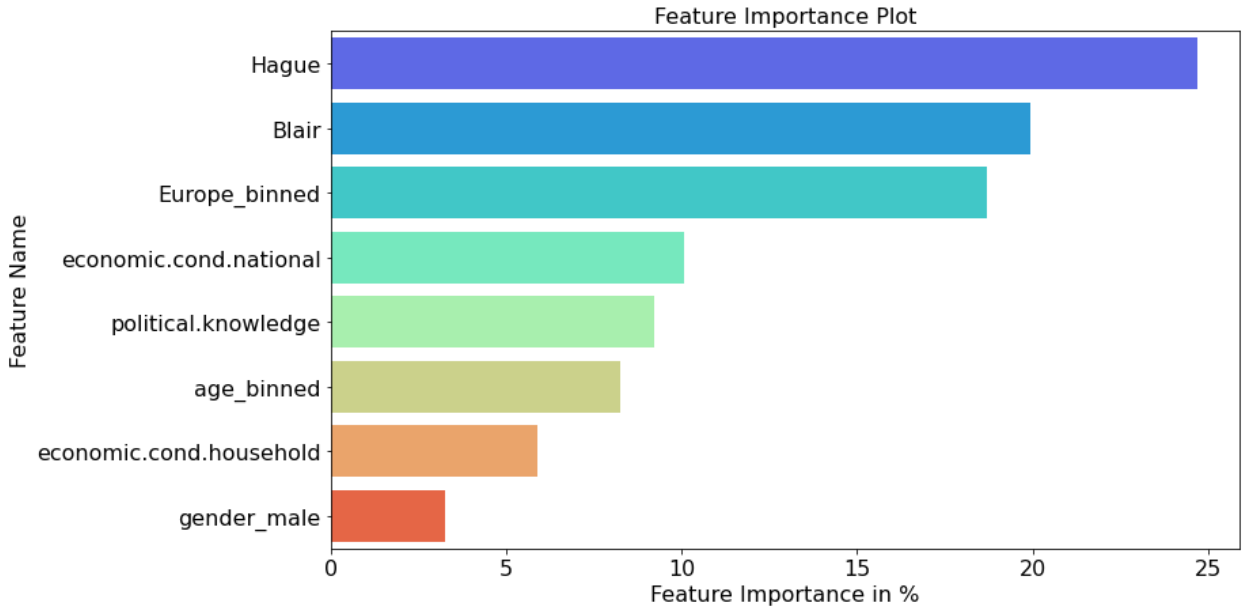


Figure 40: Feature importance as per Random Forest

LGBMClassifier:

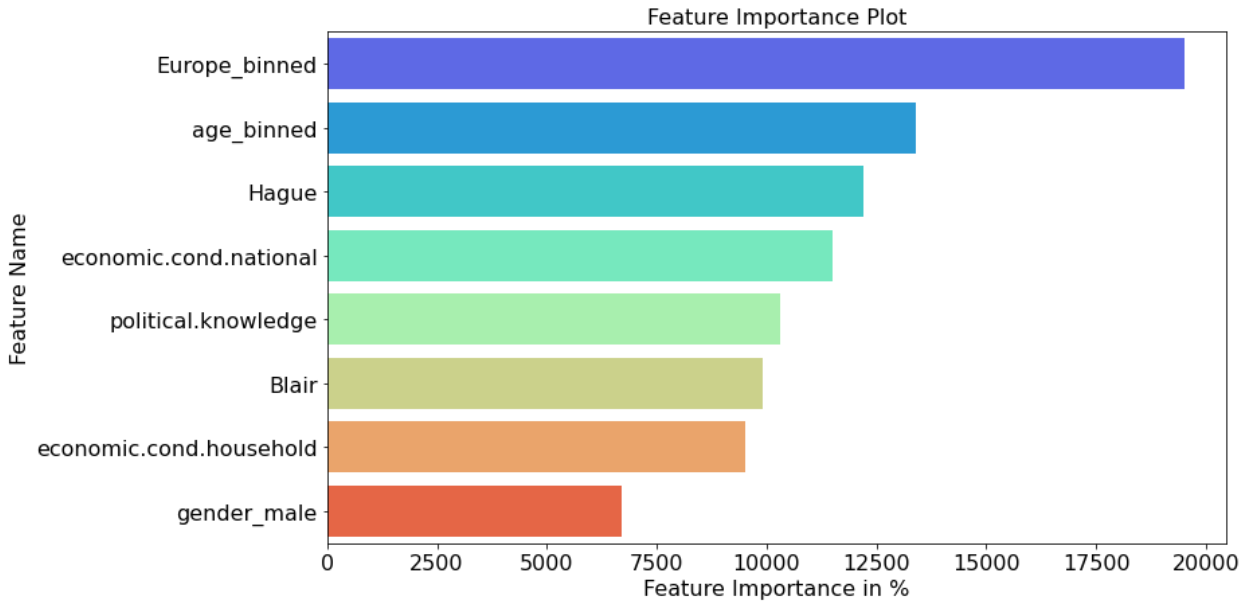


Figure 41: Feature importance as per LGBMClassifier

From the EDA performed in question 1.2, we had gathered a lot of insights, and the insights and their graphs can be referred from that question. Their interpretation after combining with above feature importances can be stated as follows:

- By comparing ranking by all the three classifiers, we can see that the assessment given for respective party leader, i.e. Blair and Hague plays a crucial role in determining the voter's desirable party we had found using the plots that people who have voted for Labour party have scored pretty positively for Blair.
- Hague has pretty high assessment from people who have voted for Conservative party but seems to be pretty negatively assessed from the voters of Labour party.
- Majority of people have given highest score to Europe which shows very high "Eurosceptic" sentiments among people.
- Voters of Conservative party are likely to have higher "Eurosceptic" sentiments than voters of Labour party whereas voters of Labour party don't have much strong opinions on Europe Integration.
- Maximum votes for Labour party are in the range of around medium age bracket of around 40-50 years and for Conservative party they are in higher bracket of age around 60-70 years.
- Voters of Labour party consider the current national economic conditions to be better than voters of Conservative party
- Voters of Labour party have similar assessment of current household economic conditions to as voters of Conservative party
- The voters considered their political knowledge to be fairly good with majority of them assessing themselves as a 2 on a scale of 3. Proportionally it is the same for both party voters.
- Gender has no significant impact to the votes.

The above observations portray the bullet points for the exit poll and will help in predicting overall win and seats covered by a particular party.

Problem 2

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

Ans)

We first loaded the speeches from inaugural corpus and stored them in a dictionary called president_dict. (line [209] of notebook)

The number of characters and words were checked in two ways:

1. **Before cleaning data:**

Number of characters: The .len() function was used to fetch the characters (Line[210]). It includes all the characters included spaces, special characters, punctuations etc.

- Number of characters in Roosevelt's speech before cleaning data using .len() method: 7571
- Number of characters in Kennedy's speech before cleaning data using .len() method: 7618
- Number of characters in Nixon's speech before cleaning data using .len() method: 9991

Number of words: The .words() and .split() method both were used to check number of words. But before cleaning data, .words() method also considered the special characters as special words (line [211]) so .split() function was used to check the number of words. It splits the words based on spaces between words (line [213]).

- Number of words in Roosevelt's speech before cleaning data using .words() method: 1536
- Number of words in Kennedy's speech before cleaning data using .words() method: 1546
- Number of words in Nixon's speech before cleaning data using .words() method: 2028

- Number of words in Roosevelt's speech before cleaning data using .split() method: 1360
- Number of words in Kennedy's speech before cleaning data using .split() method: 1390
- Number of words in Nixon's speech before cleaning data using .split() method: 1819

2. **After cleaning data:**

In text preprocessing to count the number of characters and words we created a user defined function called **clean_text** and ran it on the dictionary containing the text files which first converts all the alphabets to lower case and then removes the punctuations, special characters present in the data (Line [217]). Stemming was not performed for this purpose as the count for characters could change since words would get stemmed but the count for words will remain same.

Number of characters: The .len() function was used to fetch the characters (Line [219]).

- Number of characters in Roosevelt's speech after cleaning data using .len() method: 7359
- Number of characters in Kennedy's speech after cleaning data using .len() method: 7414
- Number of characters in Nixon's speech after cleaning data using .len() method: 9764

Number of words: The .split() method was used to check number of words. (Line [220]).

- Number of words in Roosevelt's speech after cleaning data using .split() method: 1338
- Number of words in Kennedy's speech after cleaning data using .split() method: 1365
- Number of words in Nixon's speech after cleaning data using .split() method: 1802

Number of sentences: .sents() function was used to extract the number of sentences directly from corpus. It uses characters like ".", "?" to separate the sentences (Line [215]). If we remove the special characters during data cleaning then it will not be possible to know the number of sentences in the paragraph hence this was counted before processing the data and considered as the final sentence count.

- Number of sentences in Roosevelt's speech before cleaning data using .sents() method: 68
- Number of sentences in Kennedy's speech before cleaning data using .sents() method: 52
- Number of sentences in Nixon's speech before cleaning data using .sents() method: 69

2.2) Remove all the stopwords from the three speeches.

Ans)

A list of stopwords was accessed from nltk corpus of stopwords and stored in a variable. This was used to remove the words from text files. A user defined function **remove_stopwords** was created which can remove the stopwords from any given text file. This function was run on the dictionary so that all text files present in it were now devoid of stopwords. The code can be accessed from line [221] of notebook.

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)

Ans)

Top 10 words after removing stopwords from nltk corpus:

Roosevelt:

```
# Top 10 words of Roosevelt
Roosevelt_freq = pd.Series(president_dict['Roosevelt'].split())
Roosevelt_freq.value_counts()[:10]
```

```
nation       11
know         10
spirit        9
democracy     9
us            8
life          8
people        7
america       7
years         6
freedom       6
dtype: int64
```

Figure 42: Top 10 words from Roosevelt's speech after removing stopwords from nltk corpus

Kennedy:

```
# Top 10 words of Kennedy
Kennedy_freq = pd.Series(president_dict['Kennedy'].split())
Kennedy_freq.value_counts()[:10]
```

```
let         16
us          12
sides        8
world        8
new          7
pledge       7
ask          5
free         5
citizens     5
shall        5
dtype: int64
```

Figure 43: Top 10 words from Kennedy's speech after removing stopwords from nltk corpus

Nixon:

```
# Top 10 words of Nixon
Nixon_freq = pd.Series(president_dict['Nixon'].split())
Nixon_freq.value_counts()[:10]
```

```
us               26
let              22
peace            19
world            16
new              15
america          13
responsibility   11
government       10
great             9
home              9
dtype: int64
```

Figure 44: Top 10 words from Nixon's speech after removing stopwords from nltk corpus

We can see that although words like ‘let’, ‘us’, and ‘know’ are most frequent in all the 3 speeches, they don’t provide any meaning about the context of the speech. So we will be adding these words to our stopwords list and again find the top 10 words from the speeches after filtering them out.

Roosevelt:

```
# Top 10 words of Roosevelt
Roosevelt_freq = pd.Series(president_dict['Roosevelt'].split())
Roosevelt_freq.value_counts()[:10]
```

```
nation       11
democracy     9
spirit        9
life          8
people        7
america       7
years         6
freedom       6
speaks        5
mind          5
dtype: int64
```

Figure 45: Top 10 words from Roosevelt’s speech after filtering more stopwords

Kennedy:

```
# Top 10 words of Kennedy
Kennedy_freq = pd.Series(president_dict['Kennedy'].split())
Kennedy_freq.value_counts()[:10]
```

```
world       8
sides       8
new         7
pledge      7
citizens    5
ask         5
power       5
free        5
shall       5
nations     5
dtype: int64
```

Figure 46: Top 10 words from Kennedy’s speech after filtering more stopwords

Nixon:

```
# Top 10 words of Nixon
Nixon_freq = pd.Series(president_dict['Nixon'].split())
Nixon_freq.value_counts()[:10]
```

```
peace             19
world             16
new               15
america           13
responsibility    11
government        10
great              9
home               9
abroad             8
nation             8
dtype: int64
```

Figure 47: Top 10 words from Nixon’s speech after filtering more stopwords

The top three words for each president are as follows:

- **Roosevelt:** Nation, Democracy, Spirit
- **Kennedy:** World, Sides, New
- **Nixon:** Peace, World, New

New is common for both Kennedy and Nixon, by observing the top 10 words from their speeches we are able to catch the context of their speeches to a certain extent.

Checking the top 10 words after stemming the words

Roosevelt:

```
# Top 10 words of Roosevelt
Roosevelt_freq = pd.Series(president_dict_stem['Roosevelt'].split())
Roosevelt_freq.value_counts()[:10]
```

```
nation       17
life          9
democraci     9
spirit        9
peopl         9
america       8
year          7
live          7
freedom       6
human         6
dtype: int64
```

Figure 48: Top 10 words from Roosevelt's speech after filtering more stopwords and stemming

Kennedy:

```
# Top 10 words of Kennedy
Kennedy_freq = pd.Series(president_dict_stem['Kennedy'].split())
Kennedy_freq.value_counts()[:10]
```

```
power     9
world     8
side      8
nation    8
new       7
pledg     7
ask       6
peac      5
final     5
free      5
dtype: int64
```

Figure 49: Top 10 words from Kennedy's speech after filtering more stopwords and stemming

Nixon:

```
# Top 10 words of Nixon
Nixon_freq = pd.Series(president_dict_stem['Nixon'].split())
Nixon_freq.value_counts()[:10]
```

```
america    21
peac       19
world      18
respons    17
new        15
nation     15
govern     10
year        9
home        9
great       9
dtype: int64
```

Figure 50: Top 10 words from Nixon's speech after filtering more stopwords and stemming

The top three words for each president after performing stemming are as follows:

- **Roosevelt:** Nation, Life, Democraci
- **Kennedy:** Power, World, Side
- **Nixon:** America, Peace, World

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

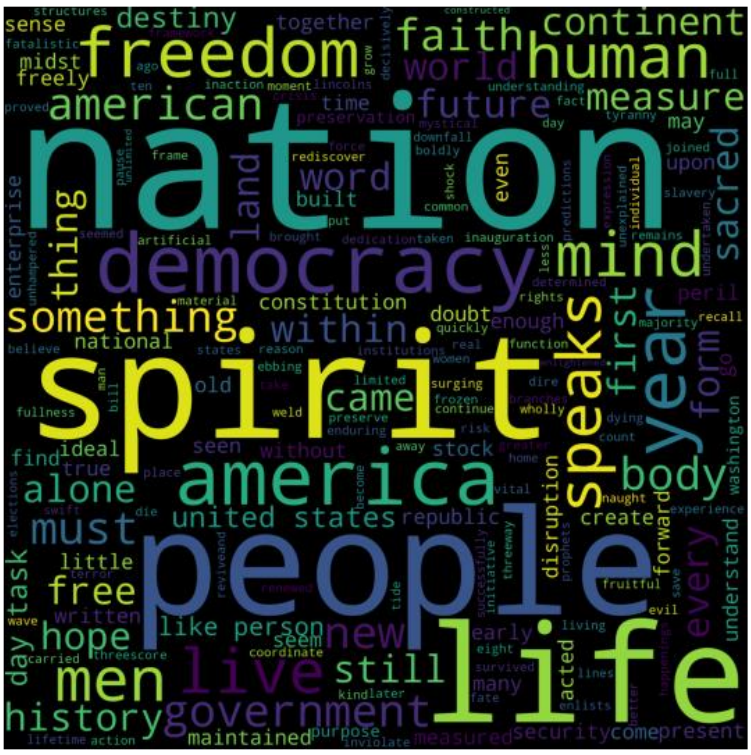


Figure 51: WordCloud for Roosevelt's speech

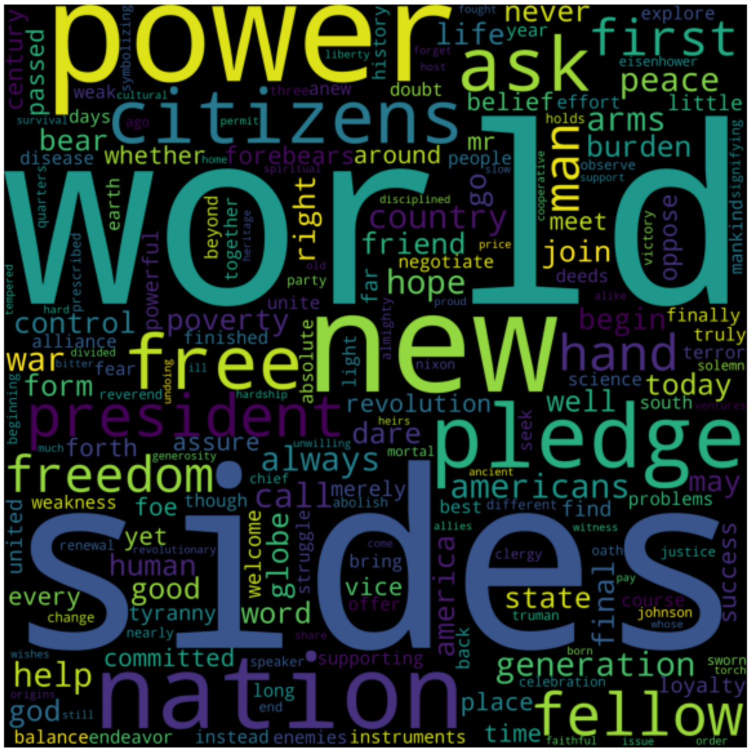


Figure 52: WordCloud for Kennedy's speech

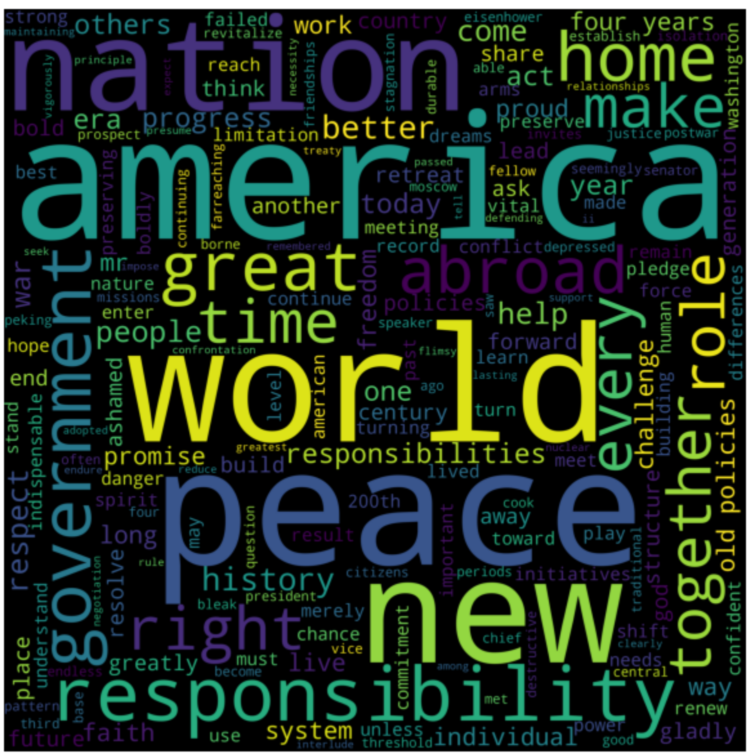


Figure 53: WordCloud for Nixon's speech