greatlearning
Power Ahead

MACHINE LEARNING

Atul Yadav
PGP-DSBA
Online
February' 23
Date: 16/08/2023

Table of Contents

Contents

THE END!

Problem 1: Machine Learning

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.

Project 1

1.1:

1.1) Read the dataset. Do the descriptive statistics and do the null value condition check.
The first step to know our data is to understand it, get familiar with it. What are the answers we're trying to get with that data? What variables are we using, and what do they mean? How does it look from a statistical perspective? Is data formatted correctly? Do we have missing values? And duplicated? What about outliers? So all these answers are can be foundout step by step as below:
Step1: Import: a) all the necessary libraries and b) The Data.
Step2: Describing the Data after loading it. Checking for datatypes, number of columns androws, checking for missing number of values, describing its min, and max, mean values. Depending upon requirement dropping off missing values or replacing it.
Step3: Reviewing new dataset and Inferences, depth and have many outliers, which can be inferred. They have three kinds of datatypes: int, float, and object.

Shape and head: The data have 1525 rows and 9 columns (we have dropped index column). vote, age, economic.cond.national, economic.cond.household, Blair, Hague, Europe, political.knowledge, gender are the 9 variables. Of which vote is dependent variable.

```
[ ] df = pd.read_excel('Election_Data.xlsx').drop('Unnamed: 0',axis=1)

[ ] df.shape

    (1525, 9)

    df.head()
```

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

+ Code + Text

Info, null values and describe: There are no null vales and dtypes are int (7) and object (2). When we check values in describe, we have see the data is quite evenly distributed. Which is again proved in skewness value, where only 'Blair' has value above -0.53.

```
[ ] df.info()

    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 1525 entries, 0 to 1524
    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: 107.4+ KB

    df.isnull().sum()

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

```
df.describe().T
```

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

```
[ ] df.skew()

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

[ ] dups = df.duplicated()
```

Duplicates: 8 duplicates value were there, which have been dropped.

```
df[dups]
```

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

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

```
df.drop_duplicates(inplace=True)
```

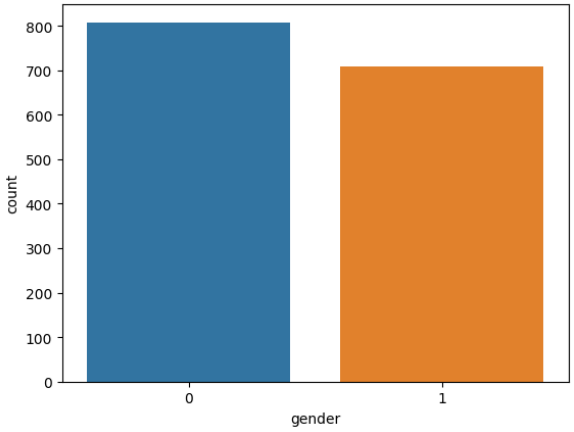
```
dups = df.duplicated()
print('Number of duplicate rows = %d' % (dups.sum()))

df[dups]
```

```
Number of duplicate rows = 0
```

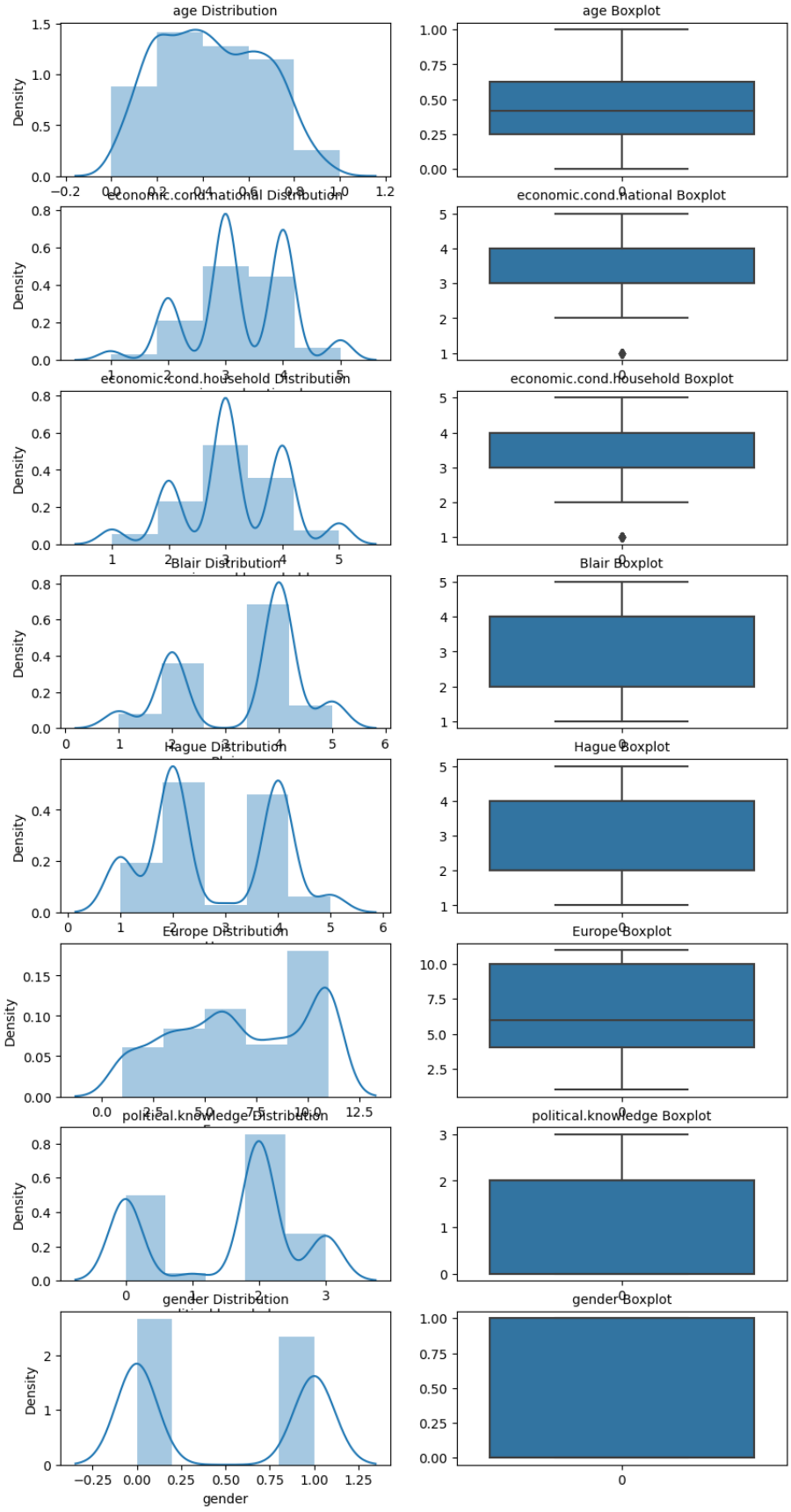
1.2). Perform Univariate and Bivariate Analysis. Do exploratory data analysis. Check for Outliers.

Univariate Analysis:

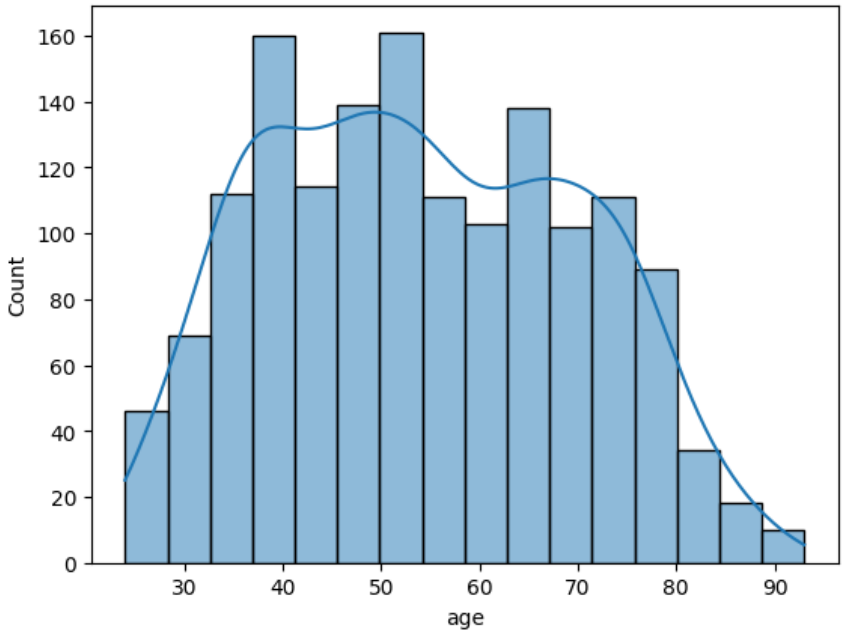
Countplot of gender: Major voting share are of female



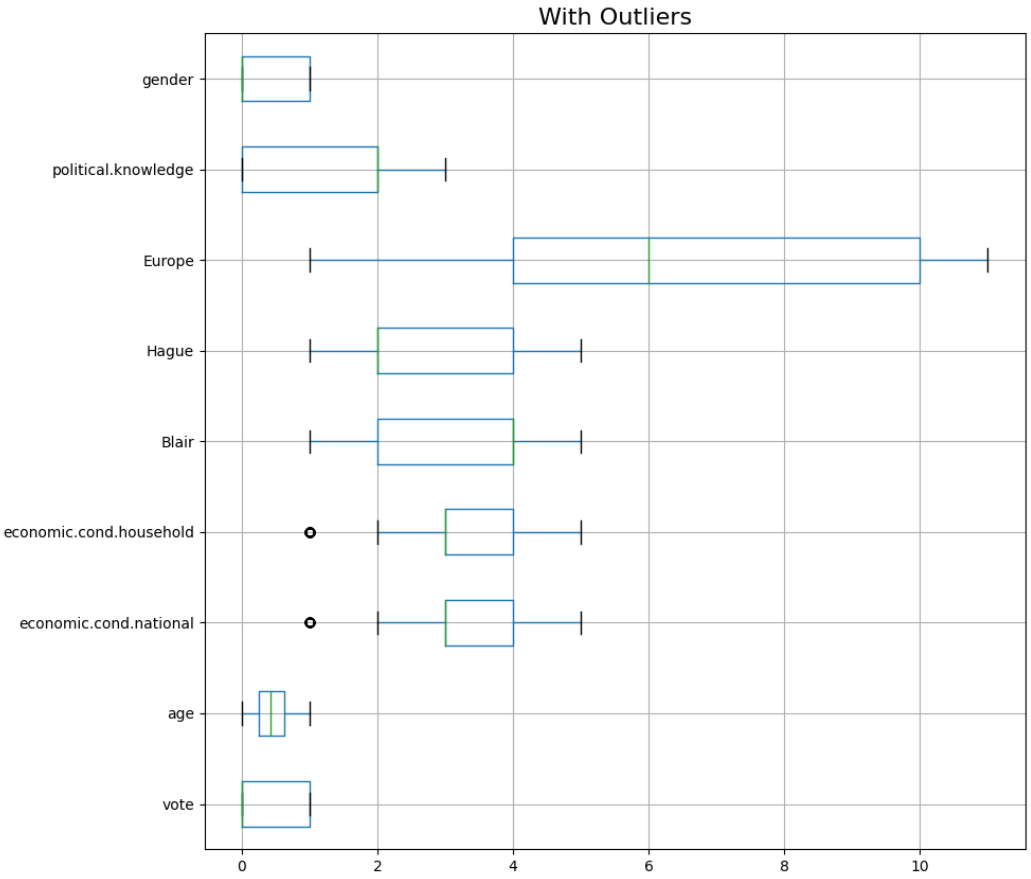
Distplot and Boxplot of all variables:



Age is a continuous variable, so we have done histplot of it as well. As expected maximum voters are of age group above 35 yrs to below 75 yrs.

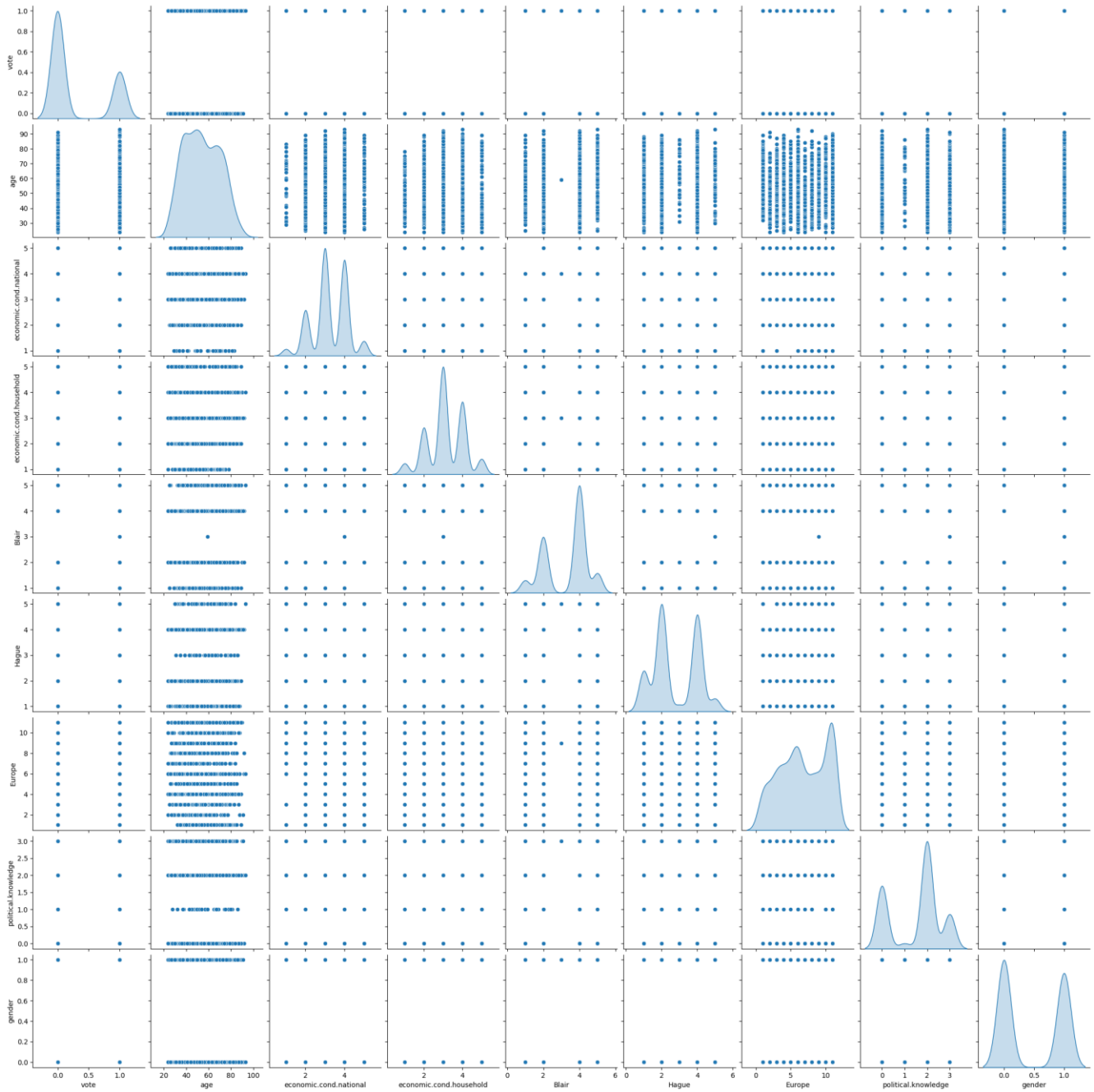


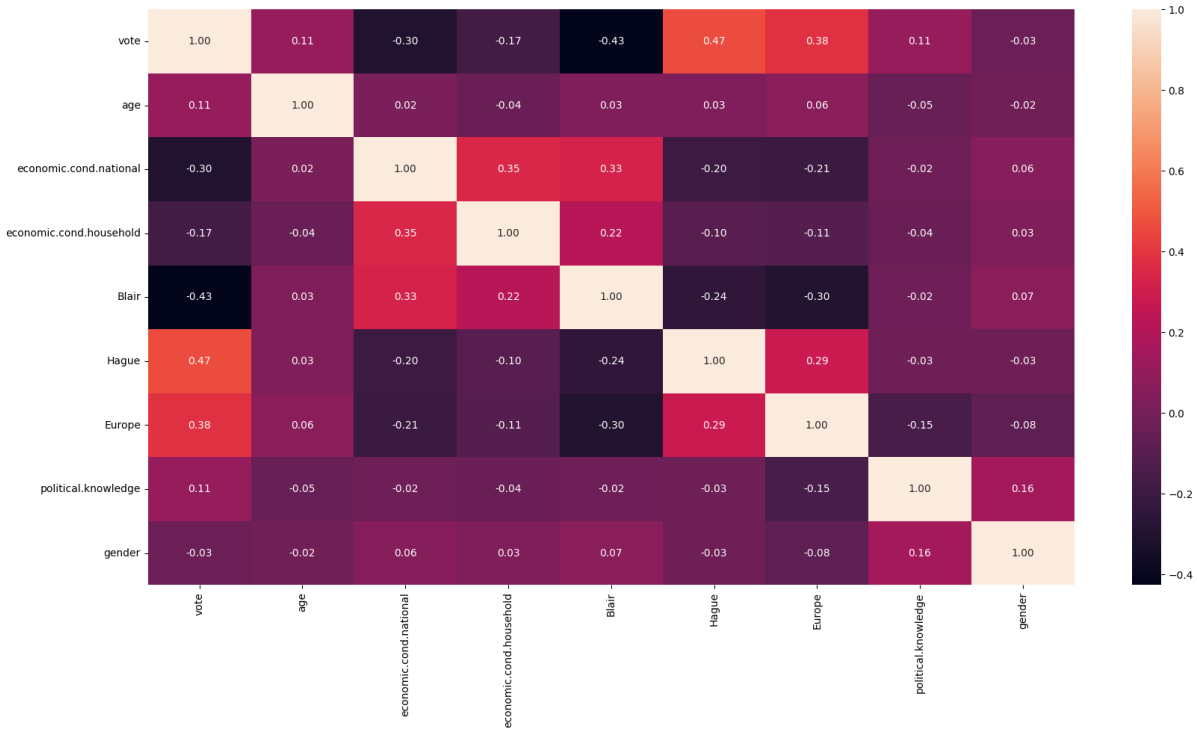
Boxplot: We can observe, outliers in economic.cond.household and economic.cond.national.



Bivariate Analysis:

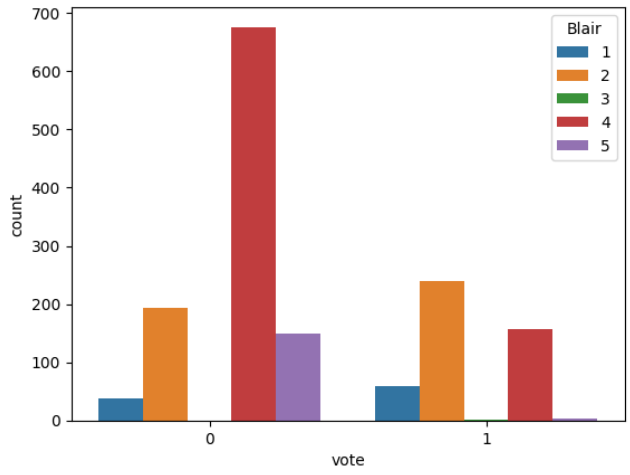
Pairplot and Heatmap: There is no correlation between any variable. Every correlation is below 0.50.

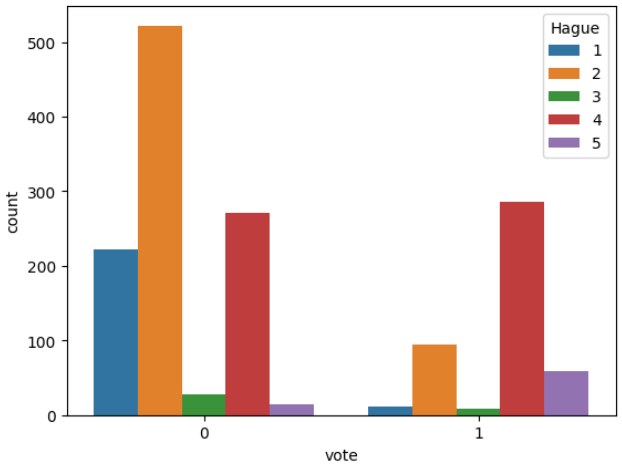




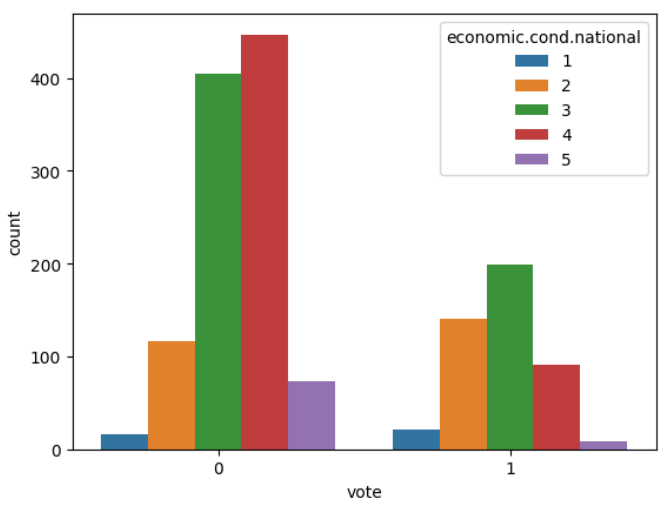
Countplot of vote with all independent variables:

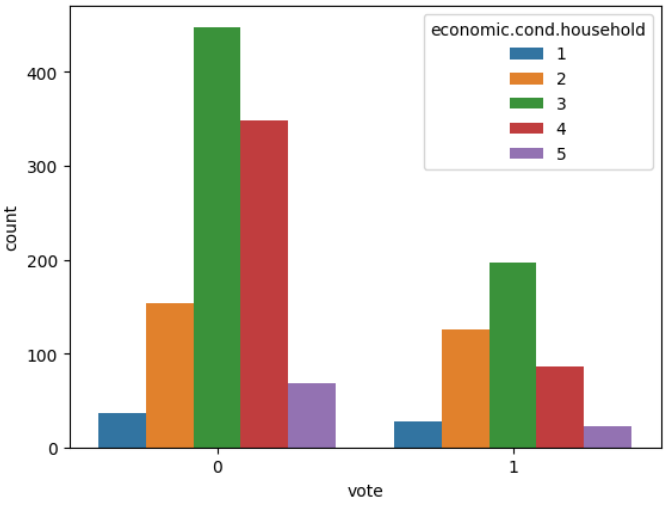
Vote with Blair and Hague: We can see a sense of dissatisfaction with Conservative leader. As voters rated Hague below 3, have strongly voted against Conservative. While voters rated Blair below 3, have almost event vote share to both parties.



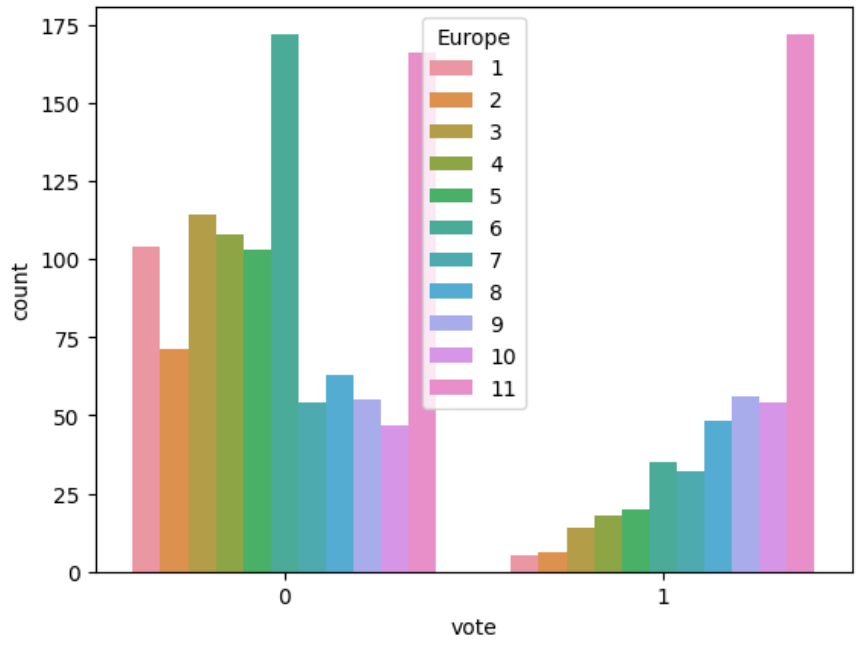


Vote with economical.cond.household and economical.cond.national: Voter with higher economic national condition and higher economic household condition have voted nore to Labour party.

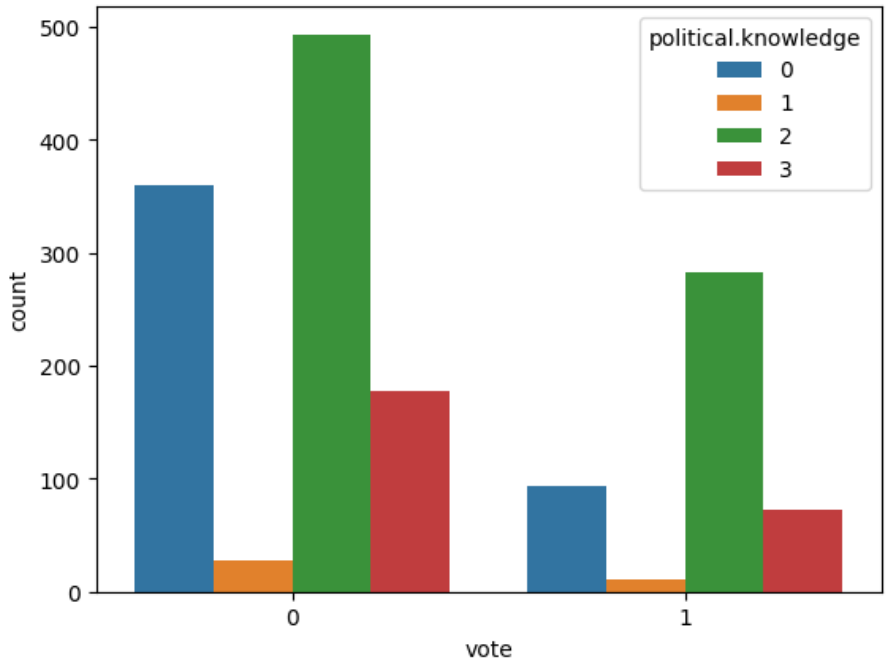




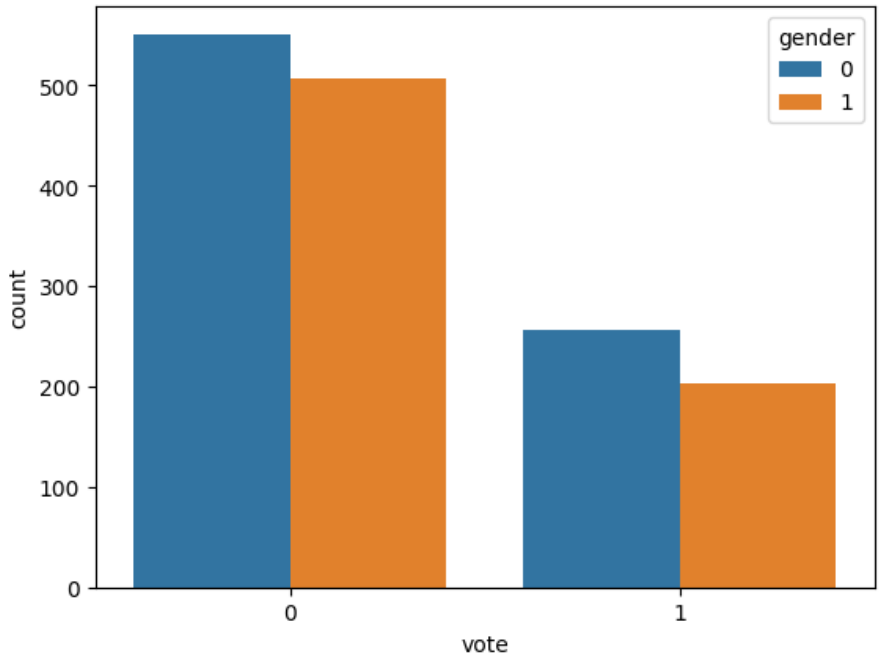
Vote and Europe: It is clearly seen, Conservative party major voter are higher Eurosceptic.



Vote and Political knowledge: Major voter of Conservative party have higher economic knowledge.



Vote and Gender:



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).

We have encoded the data. And we have scaled the data as age is continuous variable and might affect the model.

```
[ ] df[num1] = df[num1].apply(lambda x:(x-x.min()) / (x.max()-x.min()))
```

```
[ ] df.head()
```

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	0	0.275362	3	3	4	1	2	2	0
1	0	0.173913	4	4	4	4	5	2	1
2	0	0.159420	4	4	5	2	3	2	1
3	0	0.000000	4	2	2	1	4	0	0
4	0	0.246377	2	2	1	1	6	2	1

```
# independent variables
X = df.drop(["vote"], axis=1)
# dependent variable
y = df[["vote"]]
```

+ Code + Text

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=1)
```

```
[ ] stand_scal = StandardScaler()
    X_train = stand_scal.fit_transform(X_train)
    X_test = stand_scal.transform (X_test)
```

1.4. Apply Logistic Regression and LDA (linear discriminant analysis)

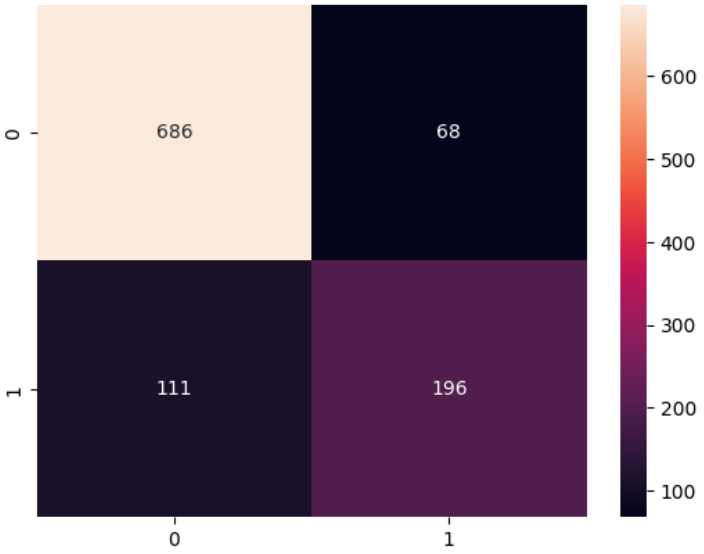
Logistic Regression:

Confusion Matrix and Classification matrix of train data:

Confusion Matrix
[[686 68]
[111 196]]

Classification Report

	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	0.77	0.79	1061
weighted avg	0.83	0.83	0.83	1061



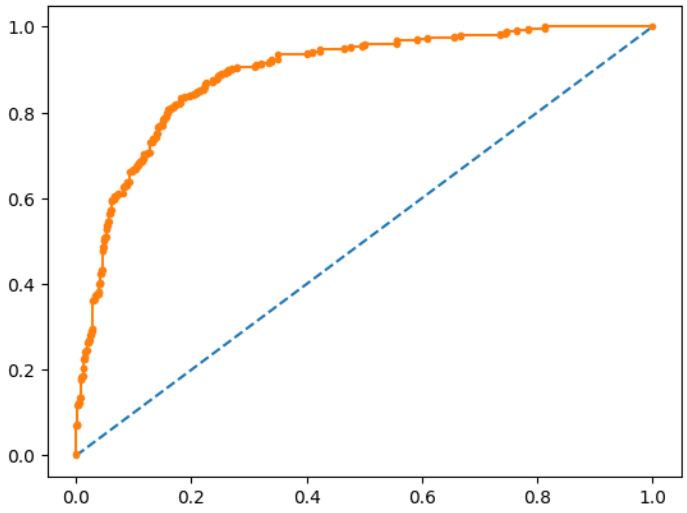
Confusion Matrix and Classification matrix of test data:

Confusion Matrix
[[268 35]
[42 111]]

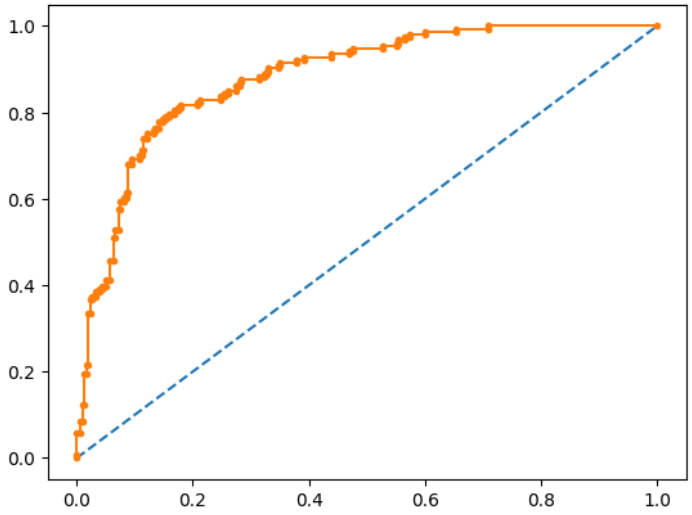
Classification Report

	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

AUC of train data: 0.890

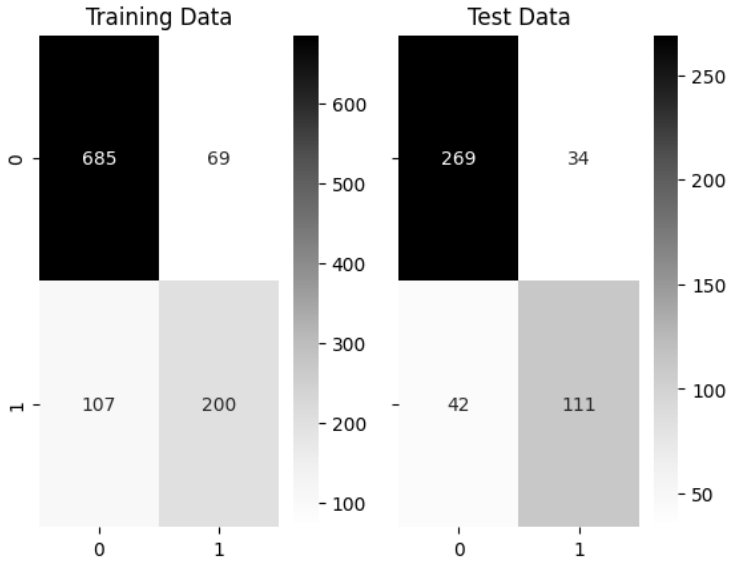


AUC of test data: 0.890



LDA:

Confusion matrix heatplot of train and test data:



Confusion Matrix and Classification matrix of train and test data:

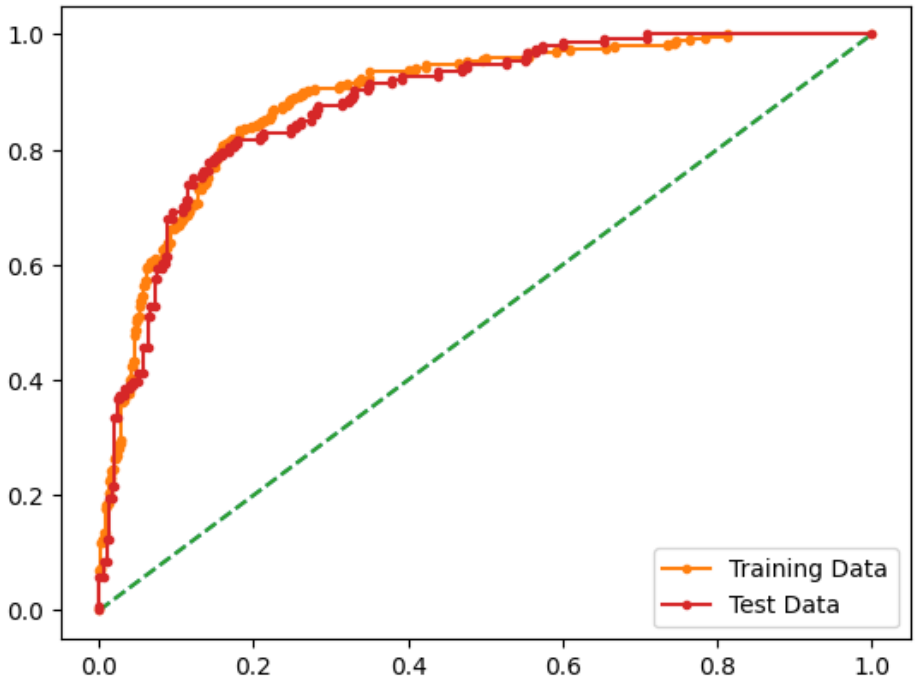
Classification Report of the training data:

	precision	recall	f1-score	support
0	0.86	0.91	0.88	754
1	0.74	0.65	0.69	307
accuracy			0.83	1061
macro avg	0.80	0.77	0.79	1061
weighted avg	0.83	0.83	0.83	1061

Classification Report of the test data:

	precision	recall	f1-score	support
0	0.86	0.88	0.88	303
1	0.77	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

AUC: train data- 0.890; test data- 0.883



1.5). Apply KNN Model and Naïve Bayes Model. Interpret the results:

KNN: Accuracy and recall value of KNN is by far highest than LDA and logistic regression. Whereas AUC value of train and test data has significant difference.

Confusion Matrix and Classification matrix of train data:

0.8539114043355325

[[689 65]
[90 217]]

	precision	recall	f1-score	support
0	0.88	0.91	0.90	754
1	0.77	0.71	0.74	307
accuracy			0.85	1061
macro avg	0.83	0.81	0.82	1061
weighted avg	0.85	0.85	0.85	1061

Confusion Matrix and Classification matrix of test data:

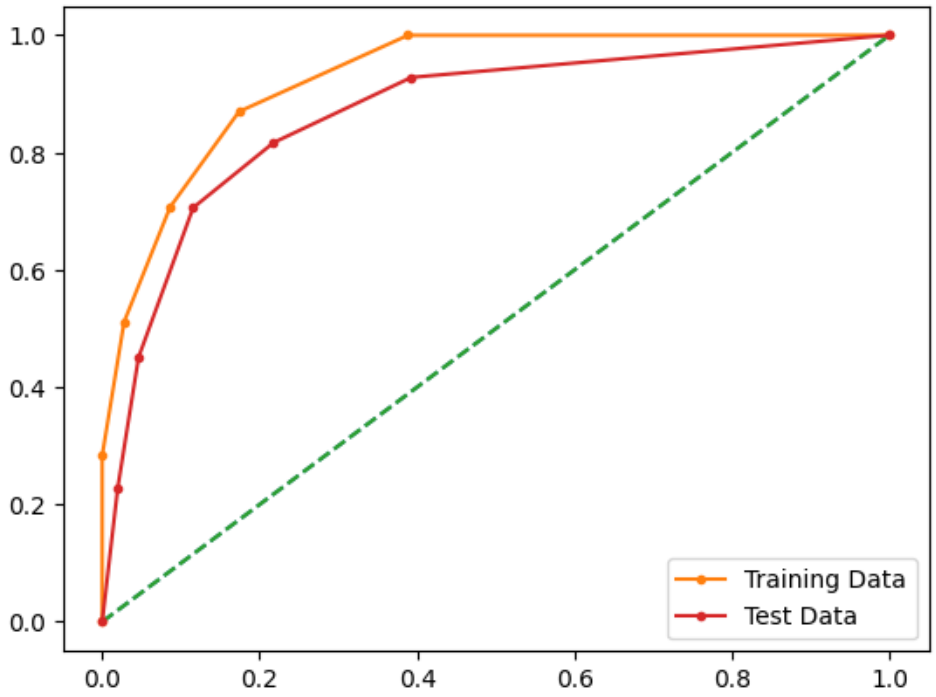
0.8245614035087719

[[268 35]

[45 108]]

	precision	recall	f1-score	support
0	0.86	0.88	0.87	303
1	0.76	0.71	0.73	153
accuracy			0.82	456
macro avg	0.81	0.80	0.80	456
weighted avg	0.82	0.82	0.82	456

AUC: train data- 0.928; test data- 0.867



Naïve Bayes: Accuracy, recall value and AUC value of Naïve Bayes is by far better than all the above models.

Confusion Matrix and Classification matrix of train data:

0.8350612629594723

[[675 79]
[96 211]]

	precision	recall	f1-score	support
0	0.88	0.90	0.89	754
1	0.73	0.69	0.71	307
accuracy			0.84	1061
macro avg	0.80	0.79	0.80	1061
weighted avg	0.83	0.84	0.83	1061

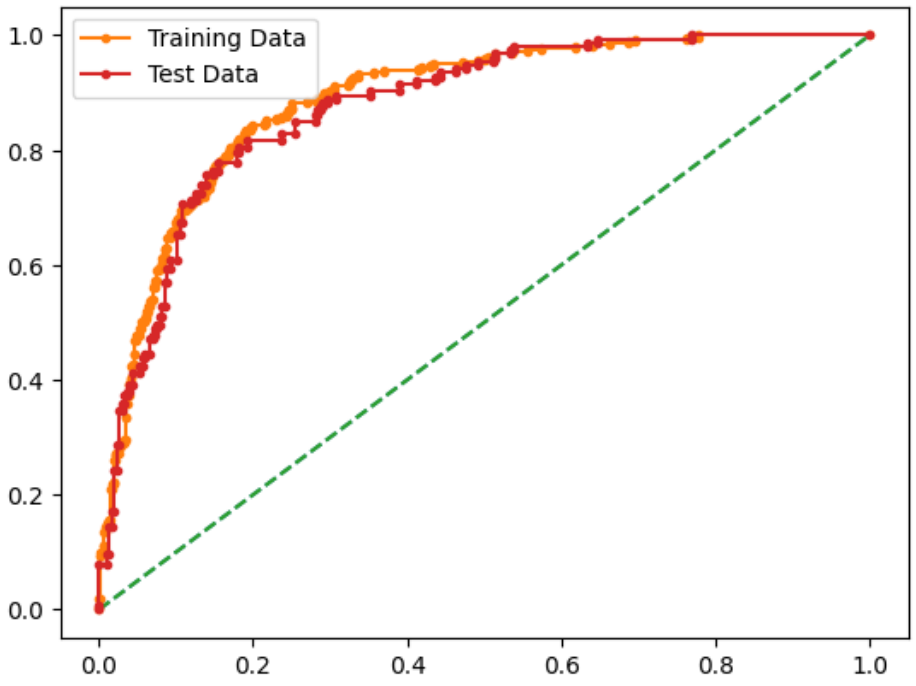
Confusion Matrix and Classification matrix of test data:

0.8223684210526315

[[263 40]
[41 112]]

	precision	recall	f1-score	support
0	0.87	0.87	0.87	303
1	0.74	0.73	0.73	153
accuracy			0.82	456
macro avg	0.80	0.80	0.80	456
weighted avg	0.82	0.82	0.82	456

AUC: train data- 0.888; test data- 0.876



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

AdaBoosting

Confusion Matrix and Classification matrix of train data:

0.8520263901979265
[[689 65]
[92 215]]

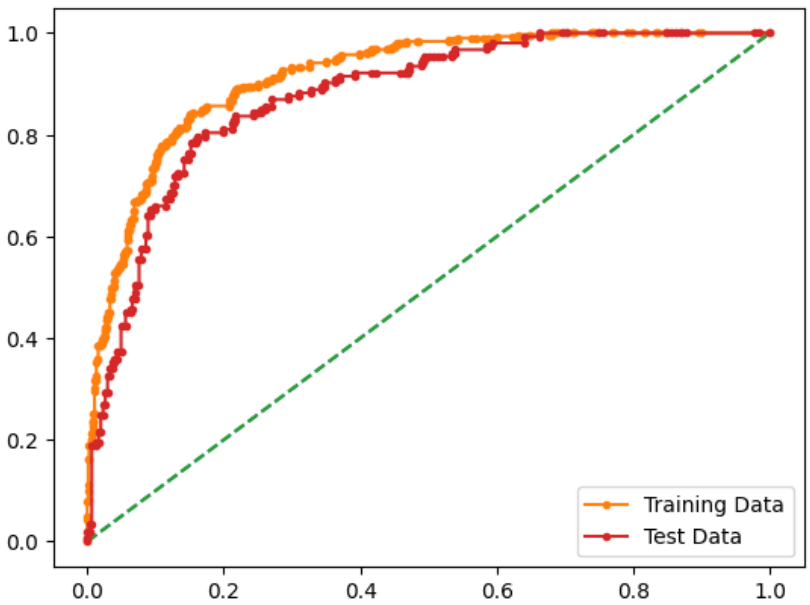
	precision	recall	f1-score	support
0	0.88	0.91	0.90	754
1	0.77	0.70	0.73	307
accuracy			0.85	1061
macro avg	0.83	0.81	0.82	1061
weighted avg	0.85	0.85	0.85	1061

Confusion Matrix and Classification matrix of test data:

0.8135964912280702
[[268 35]
[50 103]]

	precision	recall	f1-score	support
0	0.84	0.88	0.86	303
1	0.75	0.67	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

AUC: train data- 0.915; test data- 0.877



Decision Tree Classifier:

Confusion Matrix and Classification matrix of train data:

1.0
[[754 0]
[0 307]]

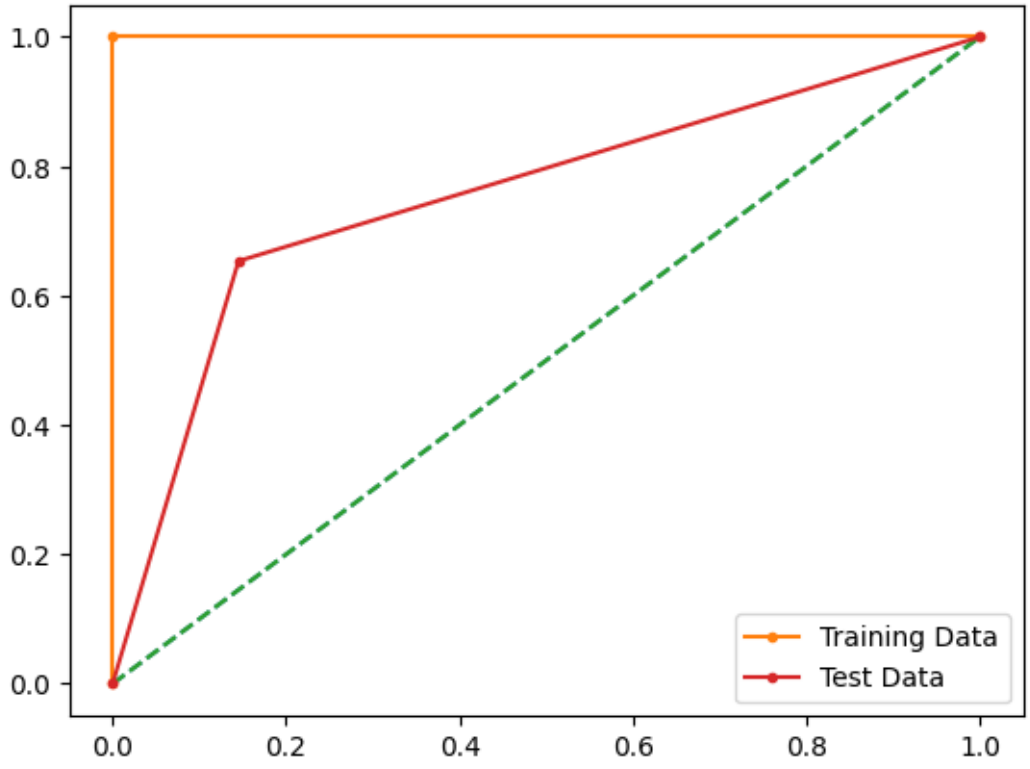
	precision	recall	f1-score	support
0	1.00	1.00	1.00	754
1	1.00	1.00	1.00	307
accuracy			1.00	1061
macro avg	1.00	1.00	1.00	1061
weighted avg	1.00	1.00	1.00	1061

Confusion Matrix and Classification matrix of train data:

0.7872807017543859
[[259 44]
[53 100]]

	precision	recall	f1-score	support
0	0.83	0.85	0.84	303
1	0.69	0.65	0.67	153
accuracy			0.79	456
macro avg	0.76	0.75	0.76	456
weighted avg	0.78	0.79	0.79	456

AUC: train data- 1.0; test data- 0.754



Random Forest Classifier:

Confusion Matrix and Classification matrix of train data:

1.0
[[754 0]
[0 307]]

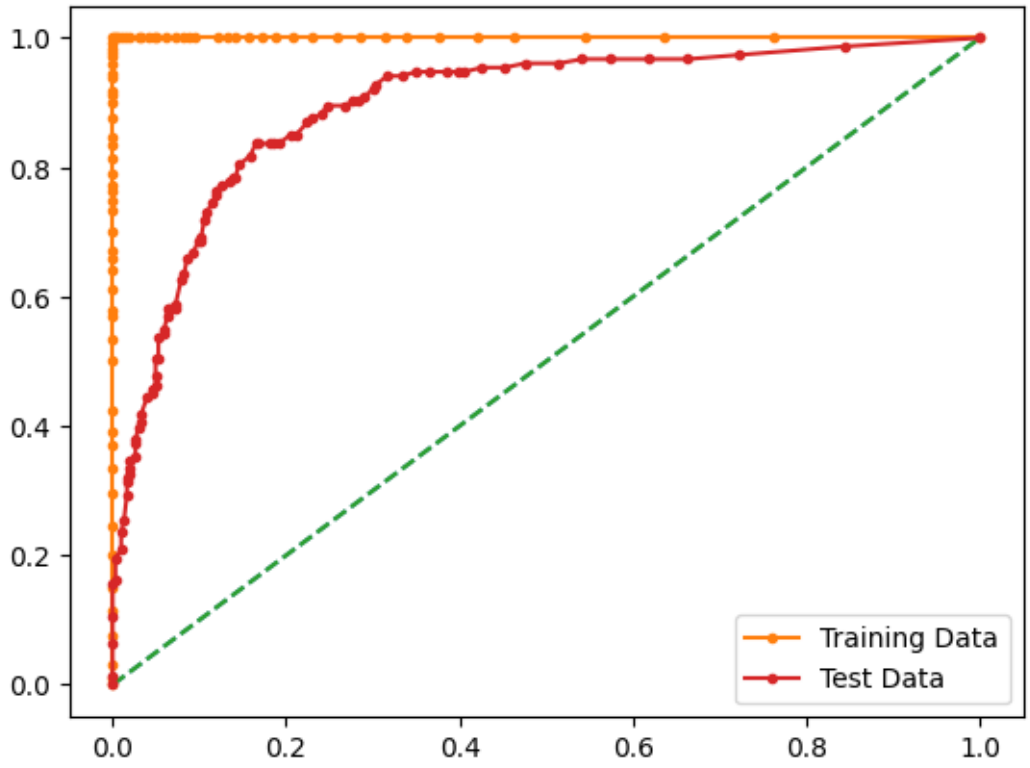
	precision	recall	f1-score	support
0	1.00	1.00	1.00	754
1	1.00	1.00	1.00	307
accuracy			1.00	1061
macro avg	1.00	1.00	1.00	1061
weighted avg	1.00	1.00	1.00	1061

Confusion Matrix and Classification matrix of test data:

0.8267543859649122
[[275 28]
[51 102]]

	precision	recall	f1-score	support
0	0.84	0.91	0.87	303
1	0.78	0.67	0.72	153
accuracy			0.83	456
macro avg	0.81	0.79	0.80	456
weighted avg	0.82	0.83	0.82	456

AUC: train data- 1.0; test data- 0.895



Random Forest Classifier:

Confusion Matrix and Classification matrix of train data:

1.0
[[754 0]
[0 307]]

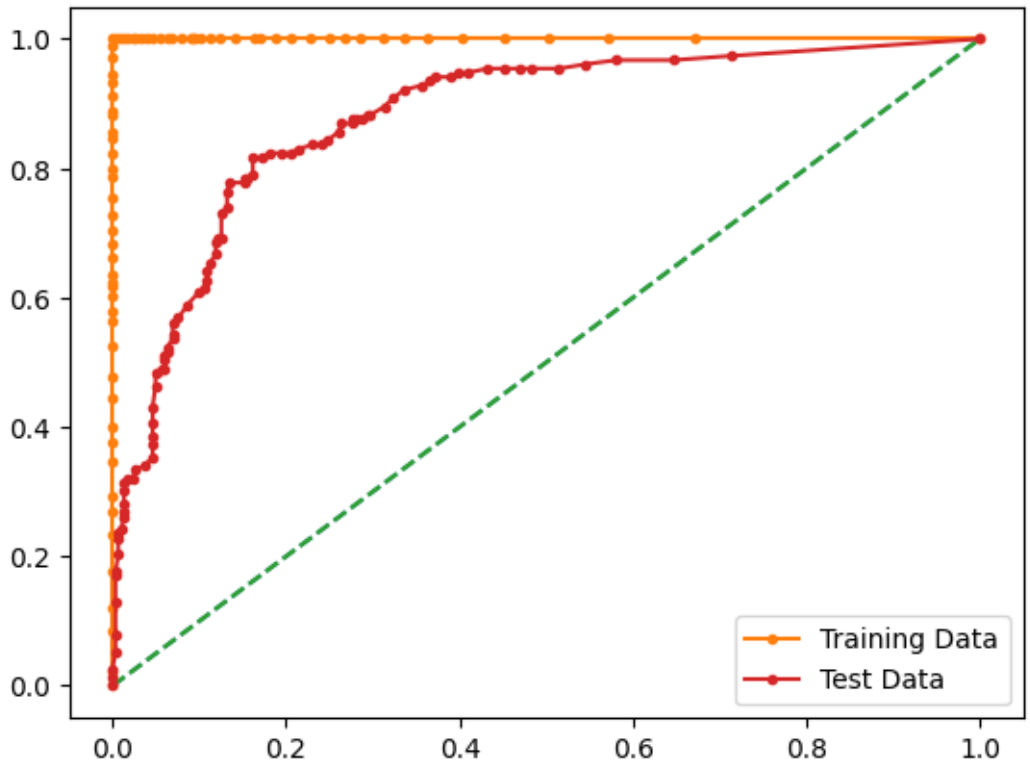
	precision	recall	f1-score	support
0	1.00	1.00	1.00	754
1	1.00	1.00	1.00	307
accuracy			1.00	1061
macro avg	1.00	1.00	1.00	1061
weighted avg	1.00	1.00	1.00	1061

Confusion Matrix and Classification matrix of test data:

0.8135964912280702
[[265 38]
[47 106]]

	precision	recall	f1-score	support
0	0.85	0.87	0.86	303
1	0.74	0.69	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

AUC: train data- 1.0; test data- 0.880



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

- We have performed performance of Predictions for each model along with model building. And after comparing the results, Naïve Bayes model is best/optimized

Model	Trained Data					Test Data				
	Accuracy	Precision	Recall	F1-Score	AUC	Accuracy	Precision	Recall	F1-Score	AUC
Logistic Regression	0.831	0- 0.86 1- 0.74	0- 0.91 1- 0.64	0- 0.88 1- 0.69	0.89	**0.83**	0- 0.86 1- 0.76	0- 0.88 1- 0.73	0- 0.87 1- 0.74	**0.89**
LDA	0.83	0- 0.86 1- 0.74	0- 0.91 1- 0.65	0- 0.88 1- 0.69	0.89	**0.83**	0- 0.86 1- 0.77	0- 0.88 1- 0.73	0- 0.88 1- 0.74	0.883
KNN	**0.853**	0- 0.88 1- 0.77	0- 0.91 1- 0.71	0- 0.90 1- 0.74	**0.92**	0.824	0- 0.86 1- 0.76	0- 0.88 1- 0.71	0- 0.87 1- 0.73	0.867
Naïve Bayes	0.84	0- 0.88 1- 0.73	0- 0.90 1- 0.69	0- 0.89 1- 0.71	0.89	0.82	0- 0.87 1- 0.74	0- 0.87 1- 0.73	0- 0.87 1- 0.73	0.876
Adaboosting	0.85	0- 0.88 1- 0.77	0- 0.91 1- 0.70	0- 0.90 1- 0.73	0.92	0.81	0- 0.84 1- 0.75	0- 0.88 1- 0.67	0- 0.86 1- 0.71	0.877
Decision tree classifier	1	0- 1 1- 1	0- 1 1- 1	0- 1 1- 1	1	0.79	0- 0.83 1- 0.69	0- 0.85 1- 0.65	0- 0.84 1- 0.67	0.754
Random forest classifier	1	0- 1 1- 1	0- 1 1- 1	0- 1 1- 1	1	**0.83**	0- 0.84 1- 0.78	0- 0.91 1- 0.67	0- 0.87 1- 0.72	**0.895**
Bagging classifier	1	0- 1 1- 1	0- 1 1- 1	0- 1 1- 1	1	0.81	0- 0.85 1- 0.74	0- 0.87 1- 0.69	0- 0.86 1- 0.71	0.88

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

1)Comparing all the performance measure, Naïve Bayes model is performing best. Although there are some other models such as KNN and AdaBoosting which is performing almost same as that of Naïve Bayes. But Naïve Bayes model is very consistent when train and test results are compared with each other. Along with other parameters such as Recall value, AUC_SCORE and AUC_ROC_Curve, those results were pretty good is this model.

2)Labour party is performing better than Conservative from huge margin.

3)Female voters turn out is greater than the male voters.

4)Those who have better national economic conditions are preferring to vote for Labour party.

5)Persons having higher Eurosceptic sentiments conservative party are preferring to vote for Conservative party.

6)Those who have higher political knowledge have voted for Conservative party

7)Looking at the assessment for both the leaders, Labour Leader is performing well as he has got better ratings in assessment.

Problem 2: Text Analytics

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

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

Code Snippet to extract the three speeches:

"

```
import nltk
nltk.download('inaugural')
from nltk.corpus import inaugural
inaugural.fileids()
inaugural.raw('1941-Roosevelt.txt')
inaugural.raw('1961-Kennedy.txt')
inaugural.raw('1973-Nixon.txt')
```

Introduction:

NLTK will provides us with everything from splitting paragraphs to sentences, splitting words, identifying the part of speech, highlighting themes, and even helping our machine understand what the text is about.

Project 2

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

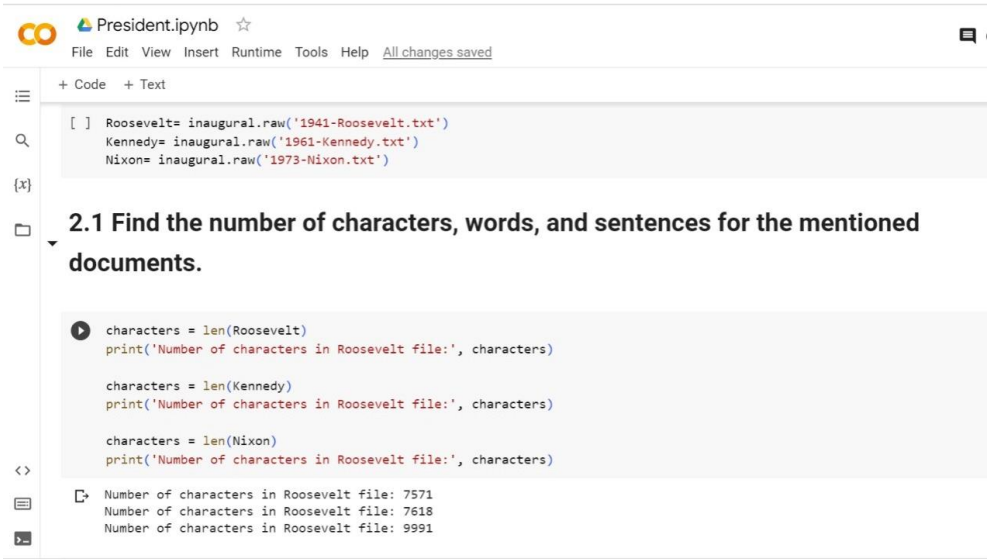
#number of Characters in each file

After importing the text file, we would first count the total number of characters in each file separately. Below is the code to count the char from each file, with the output.

Ans). Number of characters in Roosevelt file: 7571

Number of characters in Roosevelt file: 7618

Number of characters in Roosevelt file: 9991



 # Number of words in each text file:

Below we are counting the total number of words from each file.

Here we are using the split() to split up the words based on space between each word and we are counting the total number of words by using the len() function.

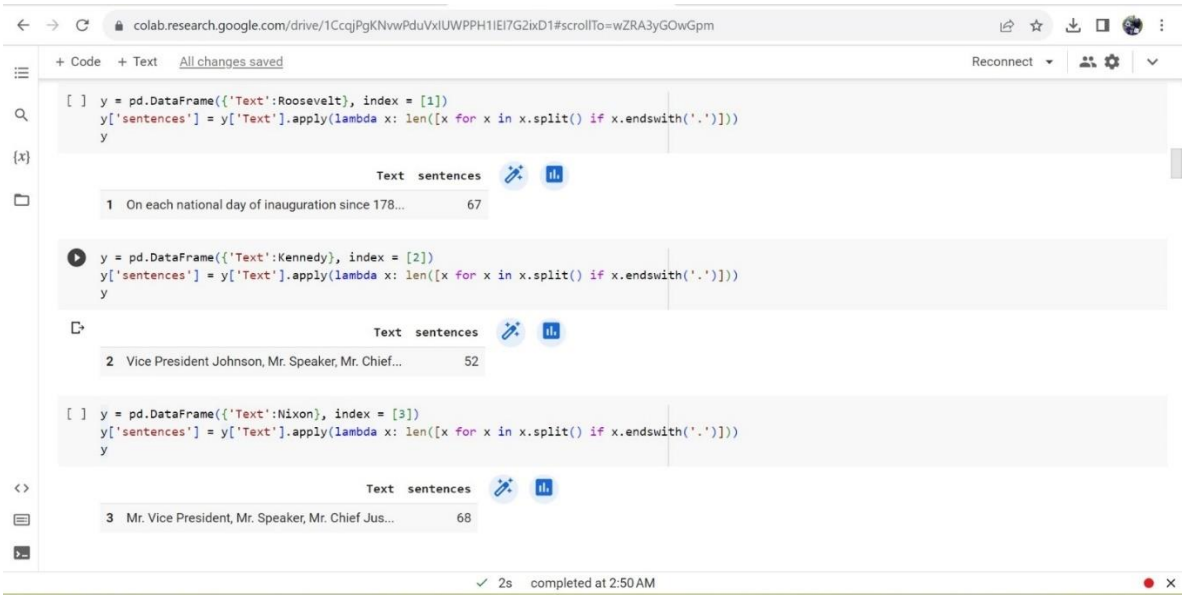
Ans). Number of words in Roosevelt file: 1360
Number of words in Kennedy file: 1390
Number of words in Nixon file: 1819

```
[ ]  words= Roosevelt.split()
     print('Number of words in Roosevelt file:', len(words))

     words= Kennedy.split()
     print('Number of words in Kennedy file:', len(words))

     words= Nixon.split()
     print('Number of words in Nixon file:', len(words))

     Number of words in Roosevelt file: 1360
     Number of words in Kennedy file: 1390
     Number of words in Nixon file: 1819
```

 # Number of Sentences.

Below we are counting the total number of sentence in each text file, by using lambda function. We are using pd.Dataframe to move the data as dictionary and then with lambda function we are checking each sentece which ends with "." Using endswith() function and the below code and output is as below.

Ans). Number of sentences in Roosevelt file: 67
Number of sentences in Kennedy file: 52
Number of sentences in Nixon file: 68

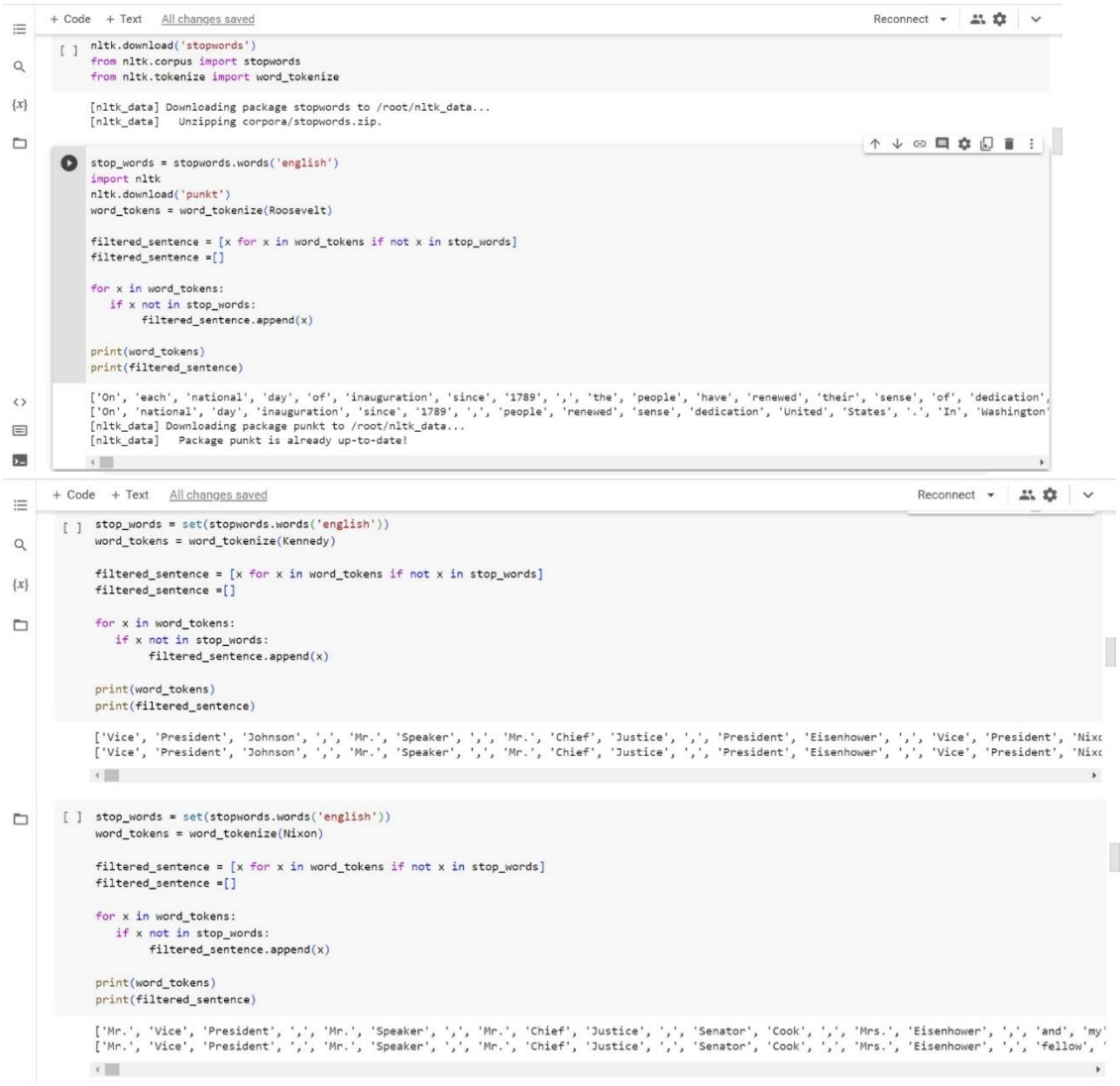


2.2. Remove all the stopwords from all three speeches

We would use the library from nltk.corpus import stopwords

from nltk.tokenize import word_tokenize. nltk.download('punkt')

We need these to remove all the English predefined words from each text file separately and with the help of tokenize we would separate each word and remove all the words from the text file.



```
[ ] nltk.download('stopwords')
    from nltk.corpus import stopwords
    from nltk.tokenize import word_tokenize

    [nltk_data] Downloading package stopwords to /root/nltk_data...
    [nltk_data]   Unzipping corpora/stopwords.zip.

    stop_words = stopwords.words('english')
    import nltk
    nltk.download('punkt')
    word_tokens = word_tokenize(Roosevelt)

    filtered_sentence = [x for x in word_tokens if not x in stop_words]
    filtered_sentence =[]

    for x in word_tokens:
       if x not in stop_words:
            filtered_sentence.append(x)

    print(word_tokens)
    print(filtered_sentence)

    ['On', 'each', 'national', 'day', 'of', 'inauguration', 'since', '1789', ',', 'the', 'people', 'have', 'renewed', 'their', 'sense', 'of', 'dedication',
    ['On', 'national', 'day', 'inauguration', 'since', '1789', ',', 'people', 'renewed', 'sense', 'dedication', 'United', 'States', '.', 'In', 'Washington'
    [nltk_data] Downloading package punkt to /root/nltk_data...
    [nltk_data]   Package punkt is already up-to-date!
```

```
[ ] stop_words = set(stopwords.words('english'))
    word_tokens = word_tokenize(Kennedy)

    filtered_sentence = [x for x in word_tokens if not x in stop_words]
    filtered_sentence =[]

    for x in word_tokens:
       if x not in stop_words:
            filtered_sentence.append(x)

    print(word_tokens)
    print(filtered_sentence)

    ['Vice', 'President', 'Johnson', ',', 'Mr.', 'Speaker', ',', 'Mr.', 'Chief', 'Justice', ',', 'President', 'Eisenhower', ',', 'Vice', 'President', 'Nixo
    ['Vice', 'President', 'Johnson', ',', 'Mr.', 'Speaker', ',', 'Mr.', 'Chief', 'Justice', ',', 'President', 'Eisenhower', ',', 'Vice', 'President', 'Nixo
```

```
[ ] stop_words = set(stopwords.words('english'))
    word_tokens = word_tokenize(Nixon)

    filtered_sentence = [x for x in word_tokens if not x in stop_words]
    filtered_sentence =[]

    for x in word_tokens:
       if x not in stop_words:
            filtered_sentence.append(x)

    print(word_tokens)
    print(filtered_sentence)

    ['Mr.', 'Vice', 'President', ',', 'Mr.', 'Speaker', ',', 'Mr.', 'Chief', 'Justice', ',', 'Senator', 'Cook', ',', 'Mrs.', 'Eisenhower', ',', 'and', 'my'
    ['Mr.', 'Vice', 'President', ',', 'Mr.', 'Speaker', ',', 'Mr.', 'Chief', 'Justice', ',', 'Senator', 'Cook', ',', 'Mrs.', 'Eisenhower', ',', 'fellow',
```

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)

We have already removed the stopwatch in previous code using stopwords.

Now we need to look for any word and count the total number of occurrences. We have performed wordcount function separately for all 3 files.

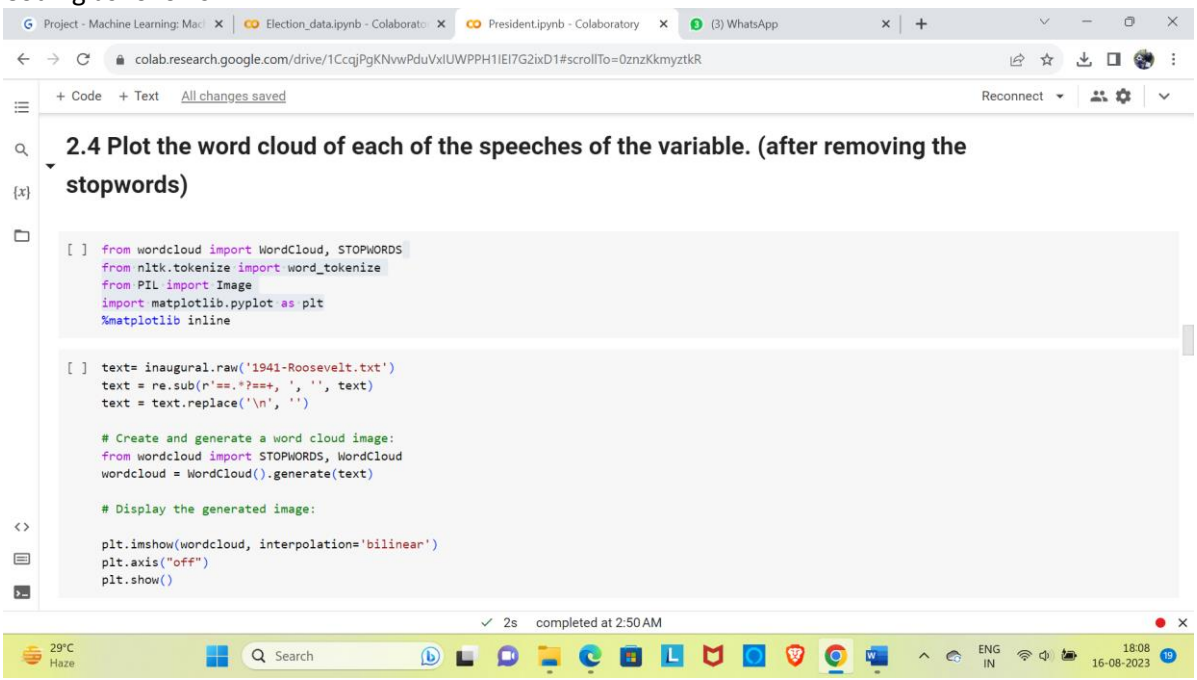
Ans). Top 3 most occurring words from inaugural address of Roosevelt file: ('Nation', 12), ('Spirit', 9), ('Life', 9)
Top 3 most occurring words from inaugural address of Kennedy file: ('World', 8), ('Sides', 8), ('Pledge', 7)
Top 3 most occurring words from inaugural address of Nixon file: ('America', 21), ('Peace', 19), ('World', 18)

```
[ ] from nltk.tokenize import RegexpTokenizer
    from collections import Counter

    tokenizer = RegexpTokenizer(r'\w+')
    roosevelt_no_punc = tokenizer.tokenize(Roosevelt)
    set(x.title() for x in roosevelt_no_punc if x.lower() not in stopwords.words())
    word_count = Counter(x.title() for x in roosevelt_no_punc if x.lower() not in stopwords.words())
    word_count.most_common(3)

    [('Nation', 12), ('Spirit', 9), ('Life', 9)]

    tokenizer = RegexpTokenizer(r'\w+')
    kennedy_no_punc = tokenizer.tokenize(Kennedy)
    set(x.title() for x in kennedy_no_punc if x.lower() not in stopwords.words())
    word_count = Counter(x.title() for x in kennedy_no_punc if x.lower() not in stopwords.words())
    word_count.most_common(3)

    [('World', 8), ('Sides', 8), ('Pledge', 7)]

[ ] tokenizer = RegexpTokenizer(r'\w+')
    nixon_no_punc = tokenizer.tokenize(Nixon)
    set(x.title() for x in nixon_no_punc if x.lower() not in stopwords.words())
    word_count = Counter(x.title() for x in nixon_no_punc if x.lower() not in stopwords.words())
    word_count.most_common(3)

    [('America', 21), ('Peace', 19), ('World', 18)]
```

2s completed at 2:50 AM

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

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. Word clouds are widely used for analysing data from social network websites.
Here we are creating the wordcloud for each speech and we have imported the wordcloud by importing libraries.
Coding as follows:



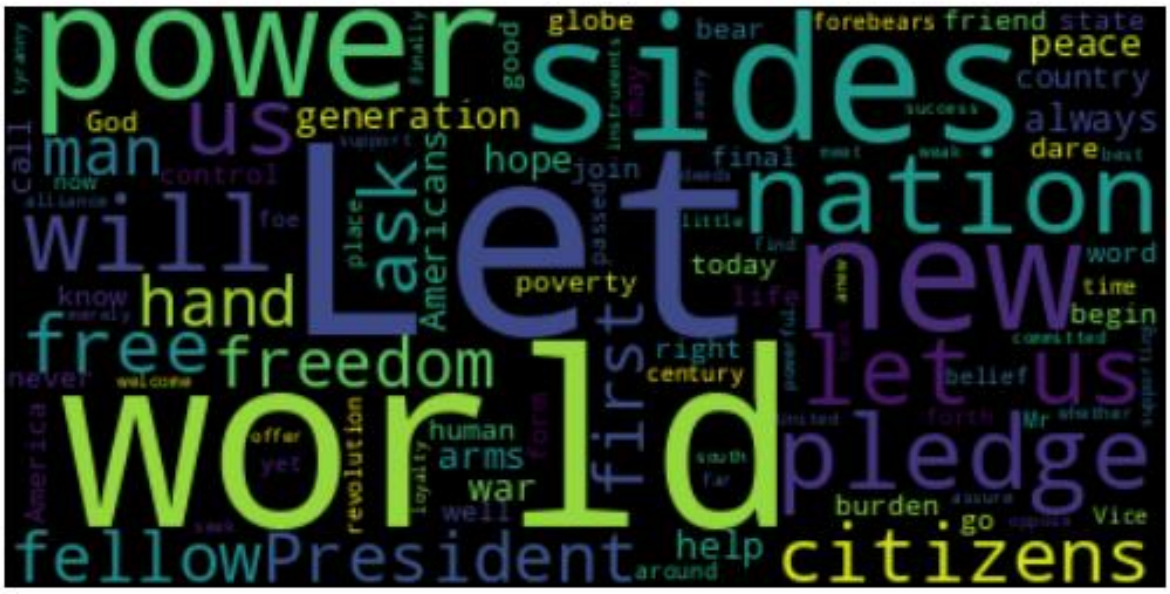
2.4 Plot the word cloud of each of the speeches of the variable. (after removing the stopwords)

```
[ ] from wordcloud import WordCloud, STOPWORDS
    from nltk.tokenize import word_tokenize
    from PIL import Image
    import matplotlib.pyplot as plt
    %matplotlib inline

[ ] text= inaugural.raw('1941-Roosevelt.txt')
    text = re.sub(r'==.*?==+, ', '', text)
    text = text.replace('\n', '')

    # Create and generate a word cloud image:
    from wordcloud import STOPWORDS, WordCloud
    wordcloud = WordCloud().generate(text)

    # Display the generated image:

    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.show()
```

2s completed at 2:50 AM

Wordcloud of Roosevelt:

Wordcloud of Kennedy:



Wordcloud of Nixon:

THE END