MACHINE LEARNING PROJECT

BUSINESS REPORT BY SHUBHANK KATAREY (PGP-DSBA)

Machine Learning

Problem 1:

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:

Vote: party choice: conservative or labour

Age: in years

economic.cond.national: assessment of current national economic conditions, 1 to 5.

economic.cond.household: assessment of current household economic conditions, 1 to 5.

Blair: assessment of the labour leader, 1 to 5.

Hague: assessment of the conservative leader, 1 to 5.

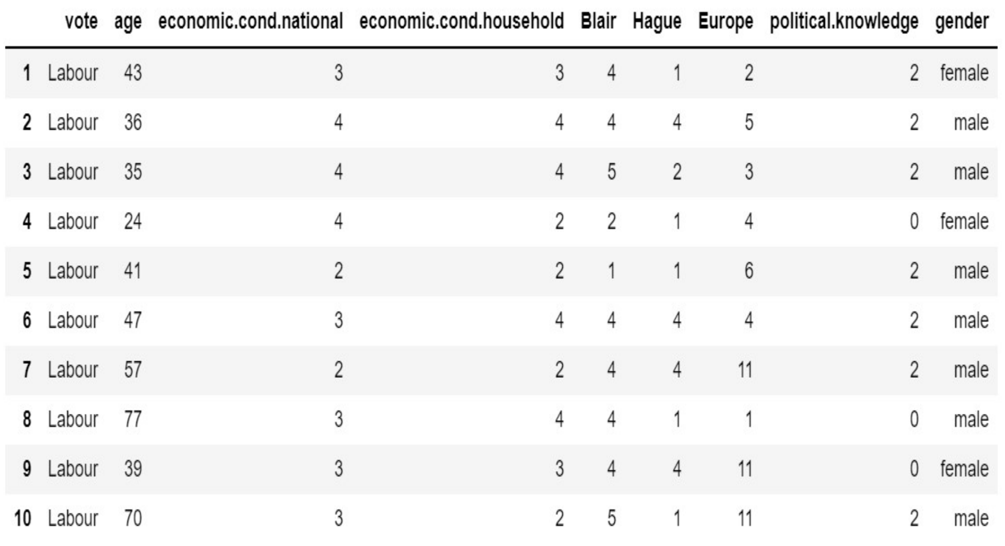
Europe: an 11-point scale that measures respondents' attitudes toward European integration. High scores represent 'eurosceptic' sentiment.

political.knowledge: knowledge of parties' positions on European integration, 0 to 3.

Gender: female or male.

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

Head of the dataset:

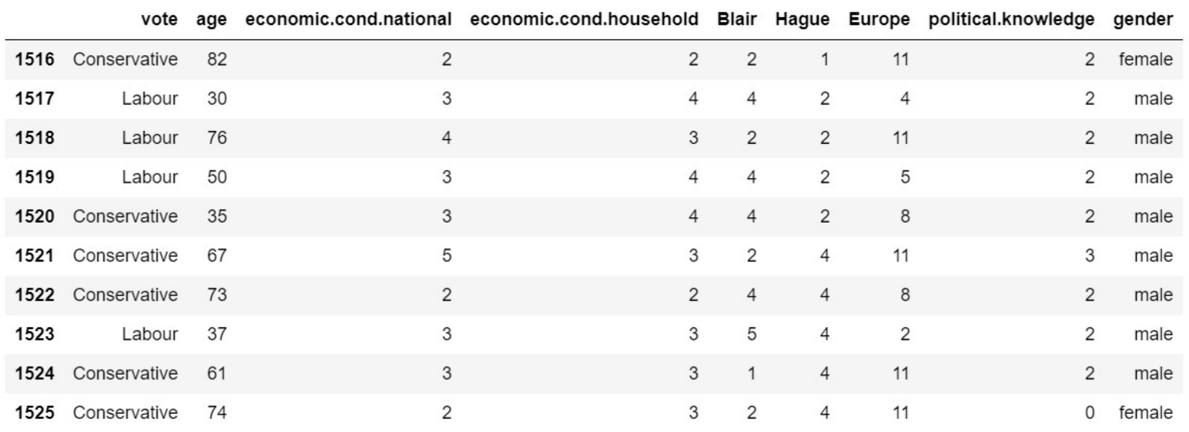


	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
1	Labour	43	3	3	4	1	2	2	female
2	Labour	36	4	4	4	4	5	2	male
3	Labour	35	4	4	5	2	3	2	male
4	Labour	24	4	2	2	1	4	0	female
5	Labour	41	2	2	1	1	6	2	male
6	Labour	47	3	4	4	4	4	2	male
7	Labour	57	2	2	4	4	11	2	male
8	Labour	77	3	4	4	1	1	0	male
9	Labour	39	3	3	4	4	11	0	female
10	Labour	70	3	2	5	1	11	2	male

The dataset is loaded properly and the top 10 rows of the dataset are displayed.

As per the given data dictionary, we can also see that the columns 'economic.cond.national' and 'economic.cond.household' are having scales from 1 to 5, so here 1 means low and 5 means good.

Tail of the dataset:



	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
1516	Conservative	82	2	2	2	1	11	2	female
1517	Labour	30	3	4	4	2	4	2	male
1518	Labour	76	4	3	2	2	11	2	male
1519	Labour	50	3	4	4	2	5	2	male
1520	Conservative	35	3	4	4	2	8	2	male
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

Shape of the dataset:

```
Number of rows: 1525
Number of columns: 9
```

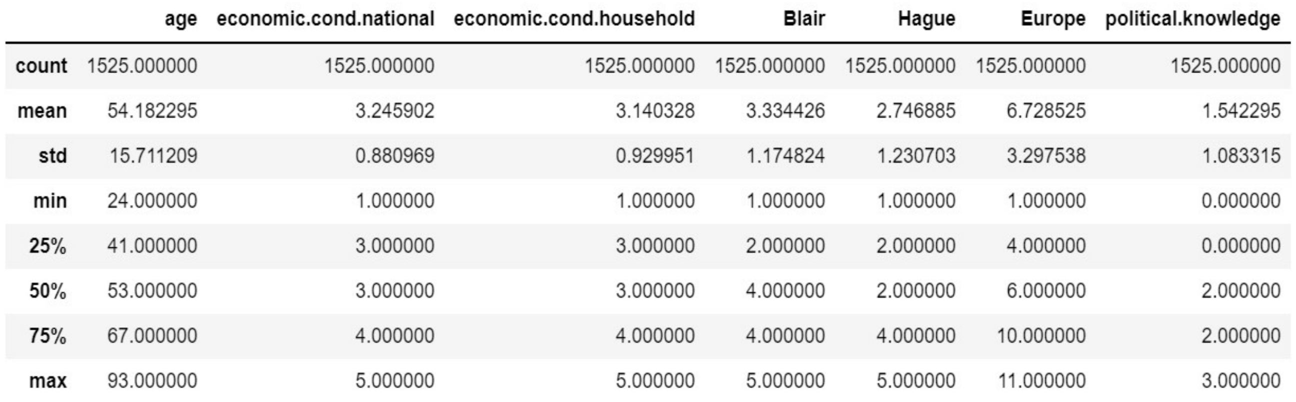
Columns in the dataset:

```
Index(['vote', 'age', 'economic.cond.national', 'economic.cond.household',
       'Blair', 'Hague', 'Europe', 'political.knowledge', 'gender'],
      dtype='object')
```

Data types in the dataset:

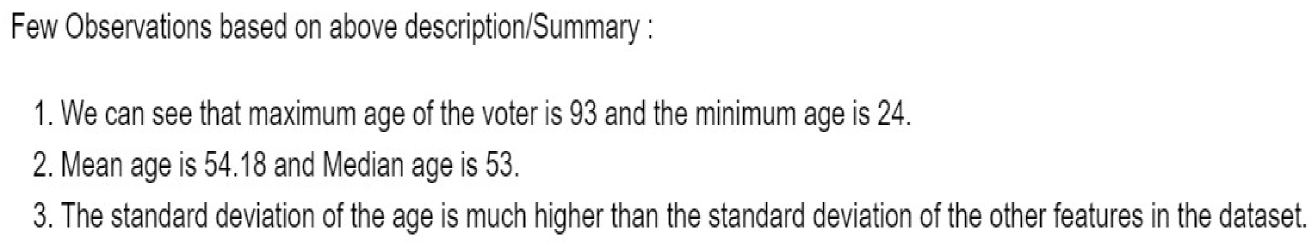
```
vote                          object
age                            int64
economic.cond.national         int64
economic.cond.household        int64
Blair                          int64
Hague                          int64
Europe                         int64
political.knowledge            int64
gender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

Description of the dataset:



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

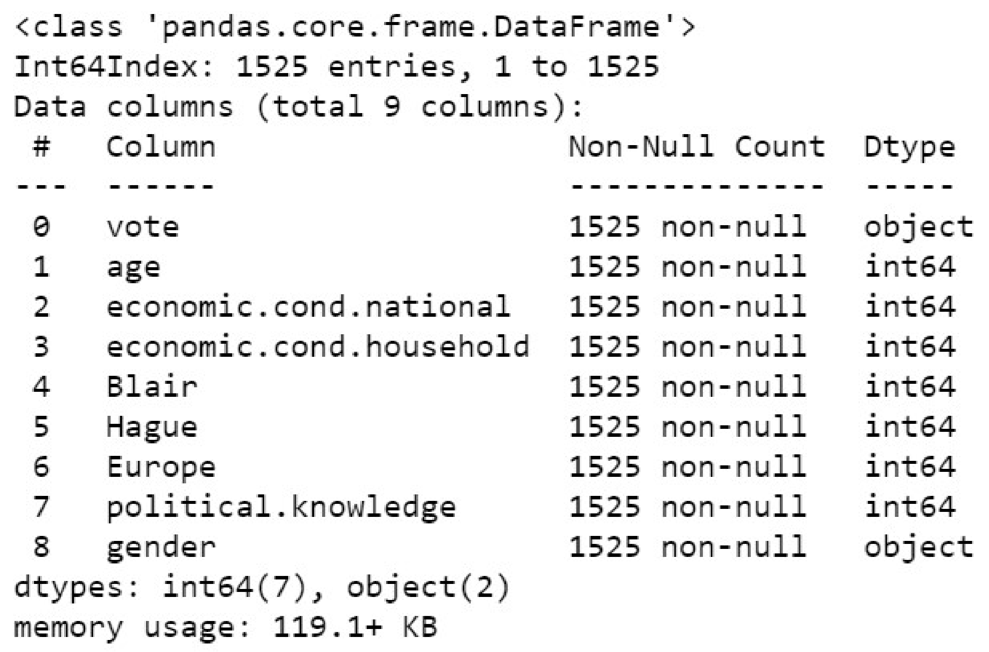
Few Observations based on Description:



Few Observations based on above description/Summary :

1. We can see that maximum age of the voter is 93 and the minimum age is 24.
2. Mean age is 54.18 and Median age is 53.
3. The standard deviation of the age is much higher than the standard deviation of the other features in the dataset.

Information about the dataset:

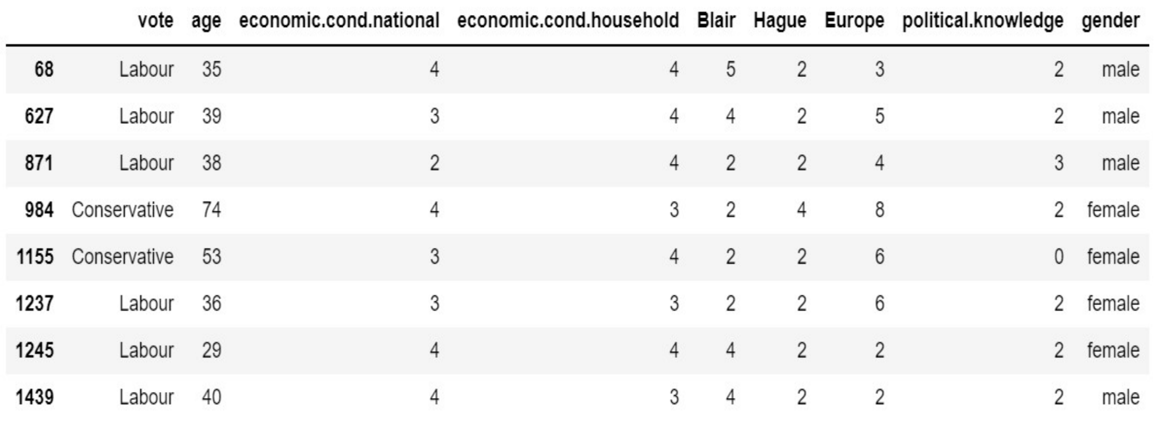


```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1525 entries, 1 to 1525
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   vote                     1525 non-null   object
 1   age                      1525 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
```

1. There are total 1525 rows and 9 columns.
2. 7 variables are of integer data types and 2 are with object data types.
3. There are no non-null values present in the dataset.

Number of Duplicates in the dataset:

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



	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
68	Labour	35	4	4	5	2	3	2	male
627	Labour	39	3	4	4	2	5	2	male
871	Labour	38	2	4	2	2	4	3	male
984	Conservative	74	4	3	2	4	8	2	female
1155	Conservative	53	3	4	2	2	6	0	female
1237	Labour	36	3	3	2	2	6	2	female
1245	Labour	29	4	4	4	2	2	2	female
1439	Labour	40	4	3	4	2	2	2	male

Shape of the dataset after removing the duplicates:

```
Number of rows: 1517
Number of columns: 9
```

Null or missing values check:

```
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
```

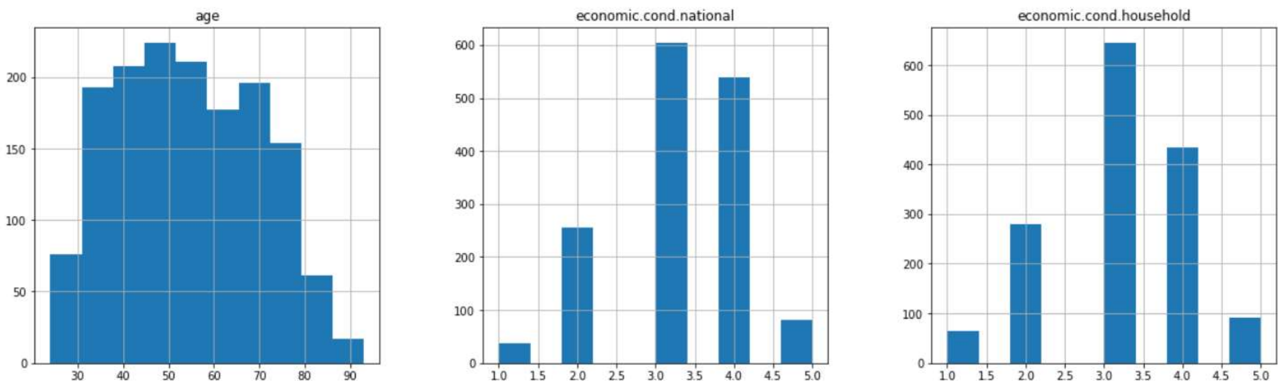
Checking vote counts:

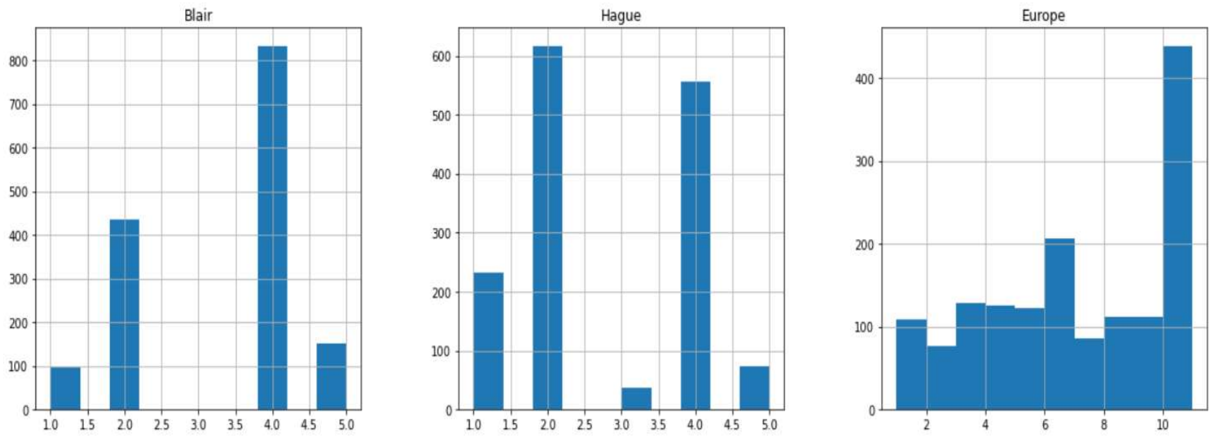
```
Labour          1057
Conservative     460
Name: vote, dtype: int64
```

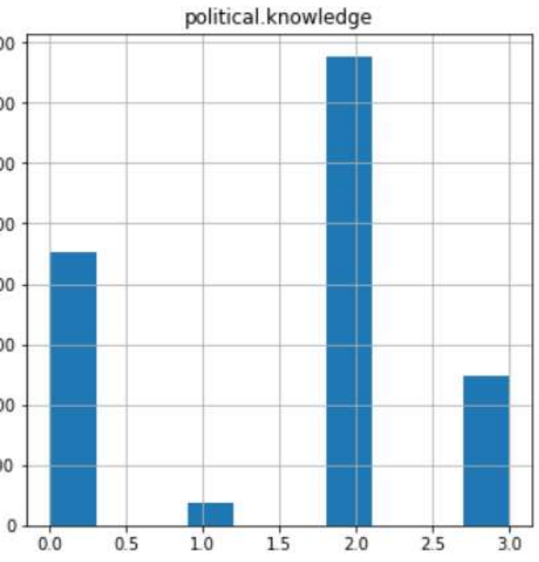
Checking gender counts:

```
female    808
male      709
Name: gender, dtype: int64
```

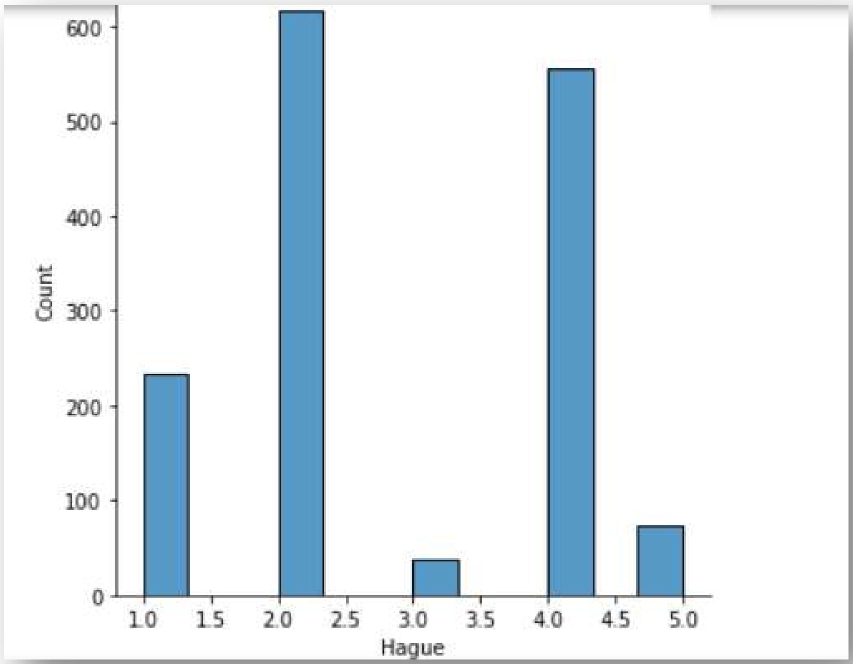
Histogram for the numeric columns in the dataset:



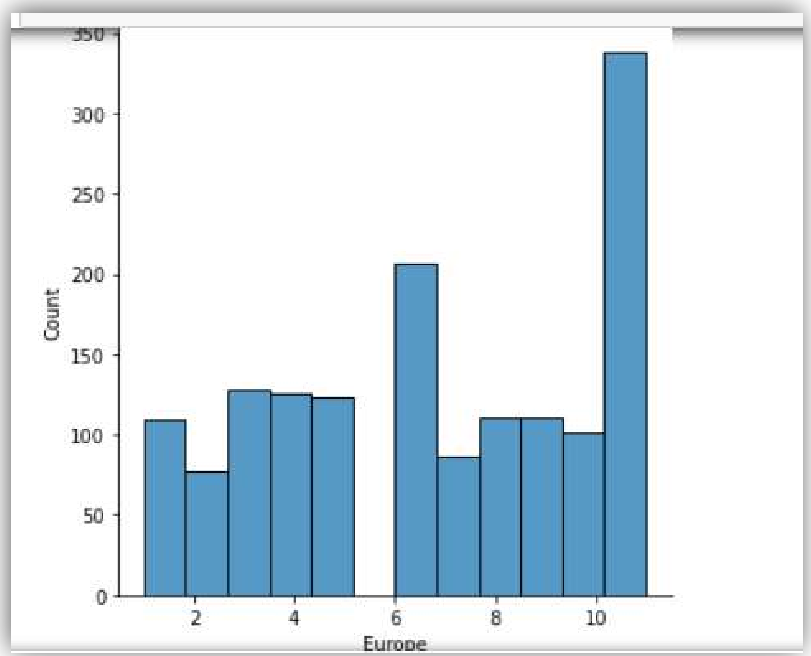




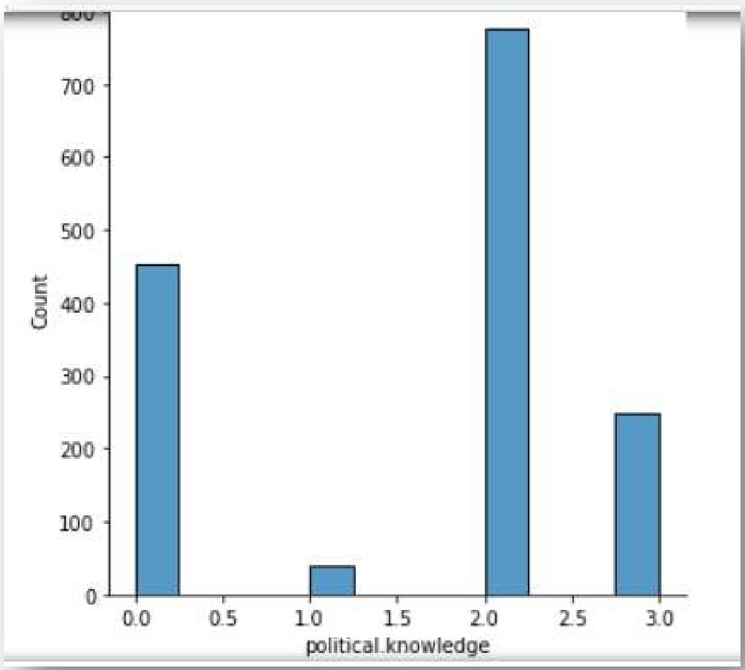
Distribution plots:



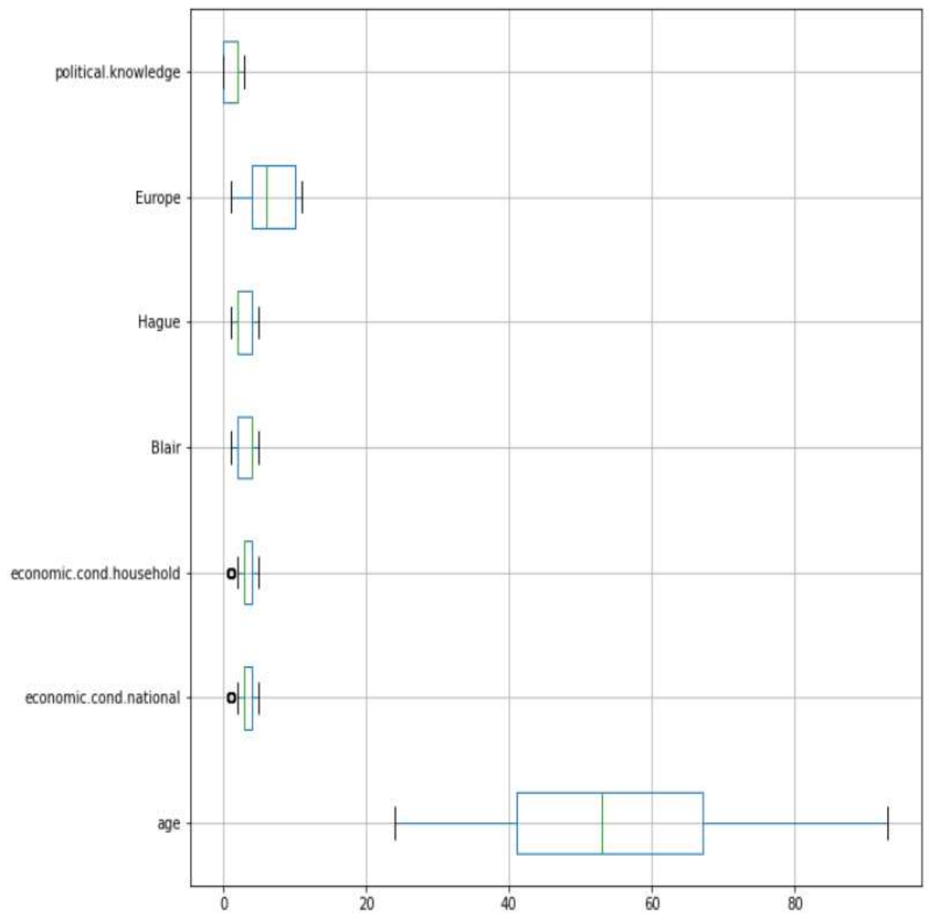
Count
Hague
0
100
200
300
400
500
600
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0



Count
Europe
0
50
100
150
200
250
300
350
2
4
6
8
10

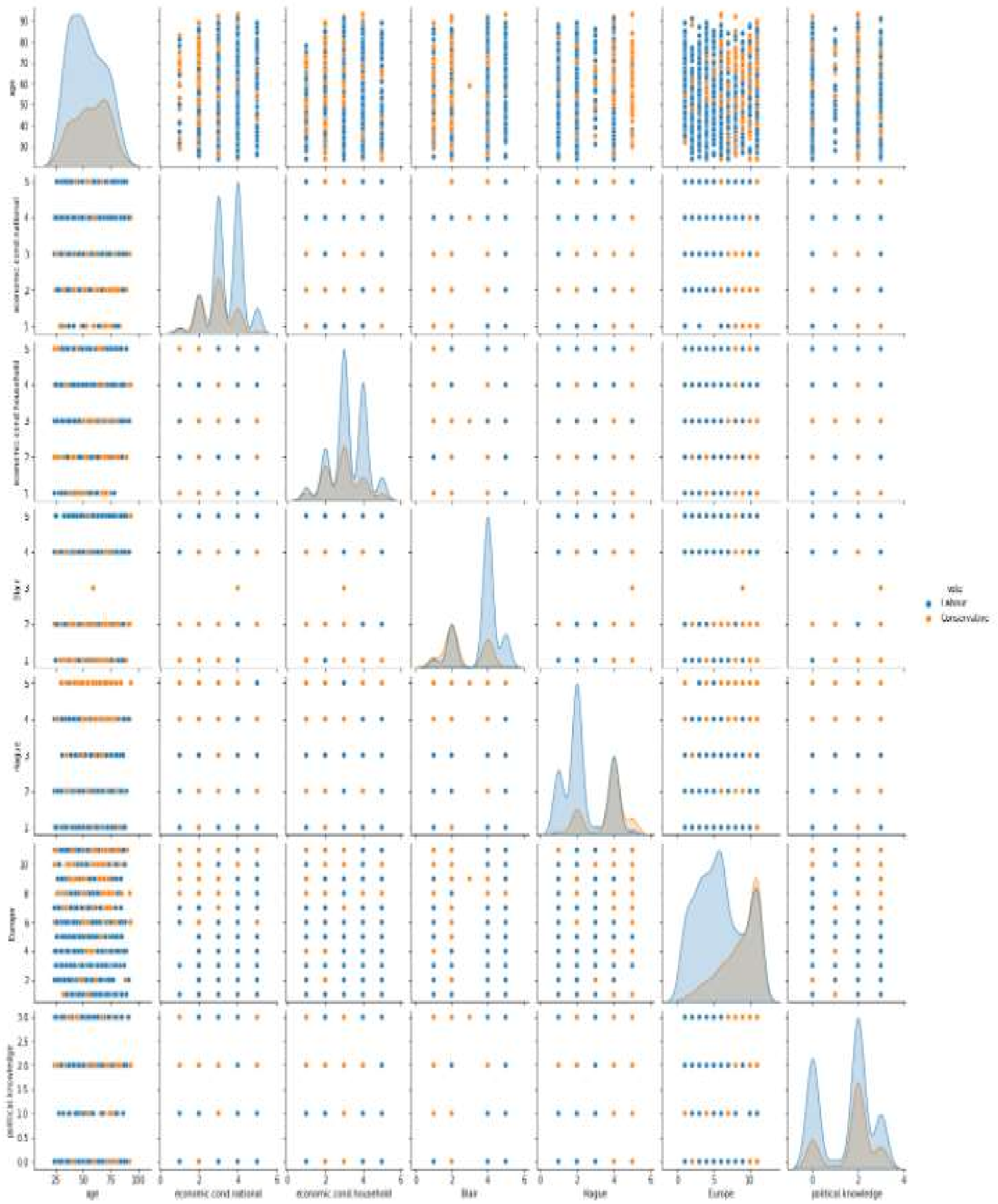


Box plot to show outliers:



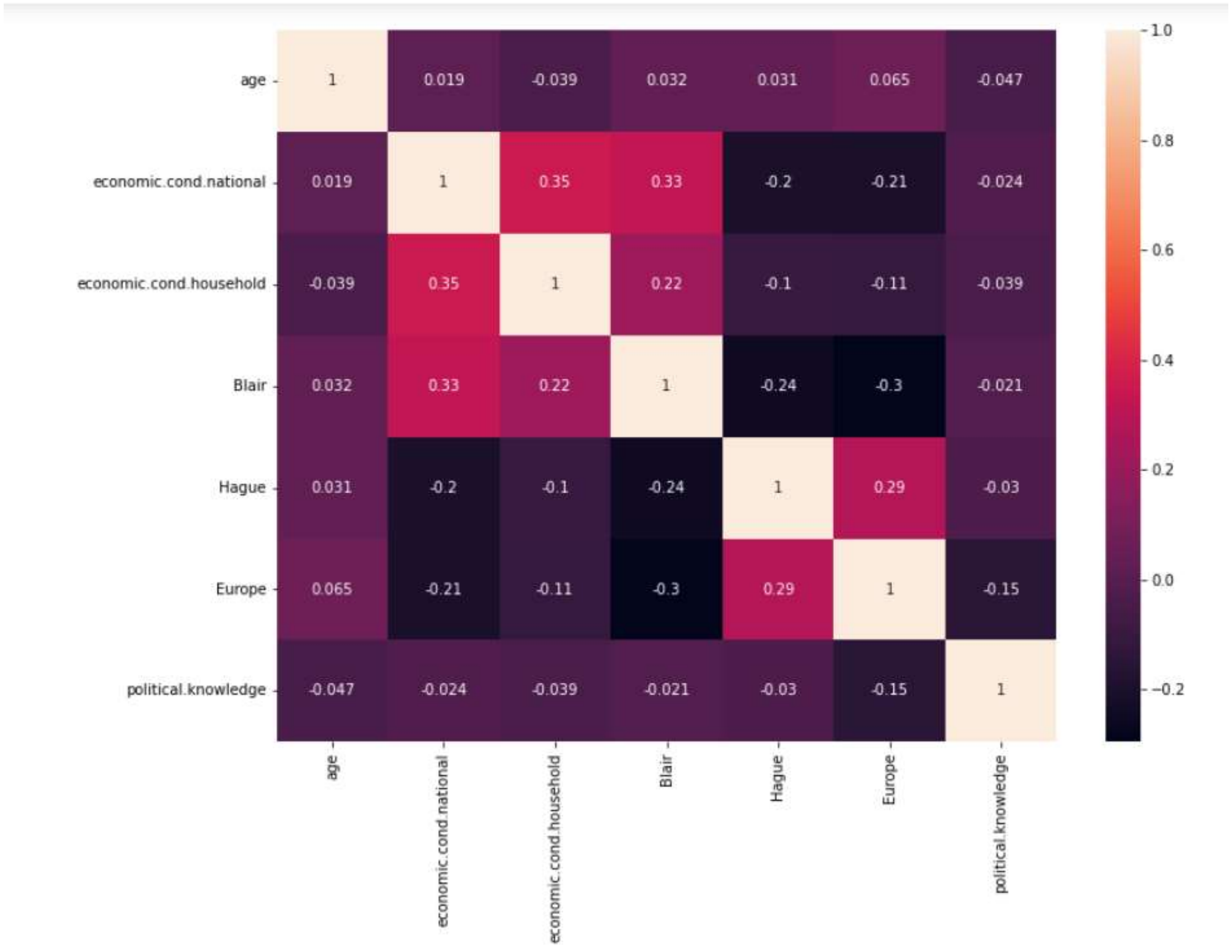
Plot is showing that very few Outliers are present in 'economic.cond.national' and 'economic.cond.household', and these are ratings or scale, so we can ignore and choose not to remove those outliers.

Pair plot:



age
economic.cond.national
economic.cond.household
Blair
Hague
Europe
political.knowledge
vote
Labour
Conservative

Heat map:

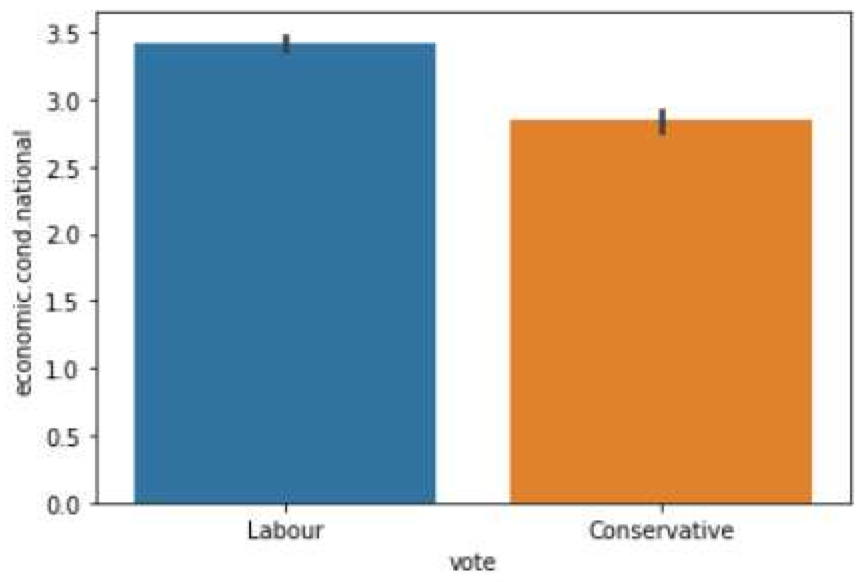


	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge
age	1	0.019	-0.039	0.032	0.031	0.065	-0.047
economic.cond.national	0.019	1	0.35	0.33	-0.2	-0.21	-0.024
economic.cond.household	-0.039	0.35	1	0.22	-0.1	-0.11	-0.039
Blair	0.032	0.33	0.22	1	-0.24	-0.3	-0.021
Hague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

Bar plot:

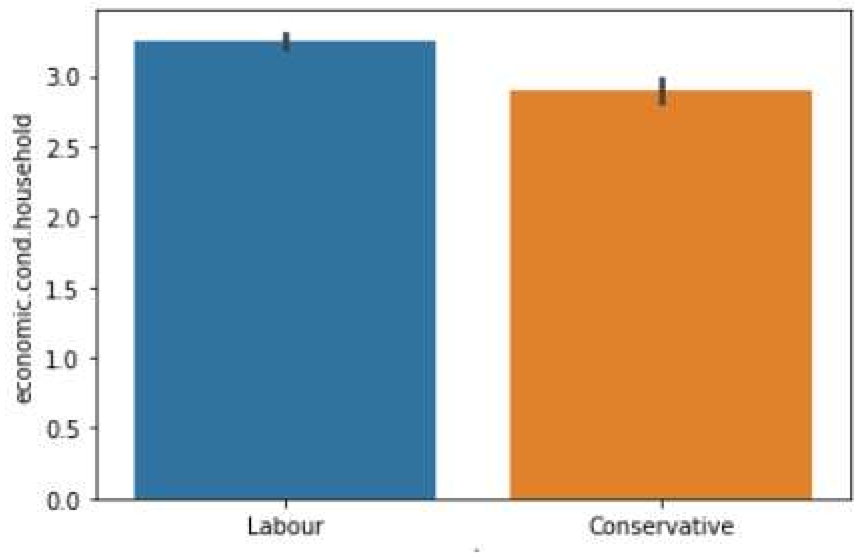
```
sns.barplot(data = df, x='vote',y='economic.cond.national')
```

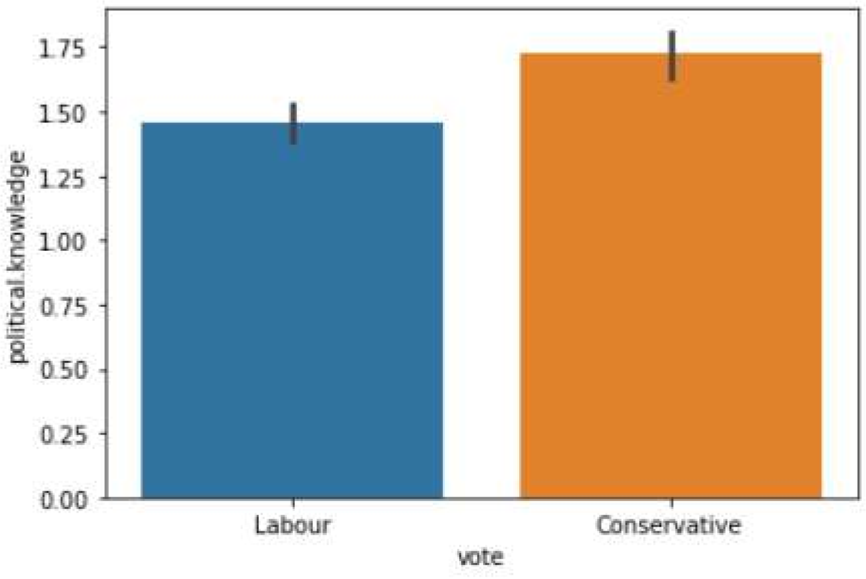
```
<AxesSubplot:xlabel='vote', ylabel='economic.cond.national'>
```



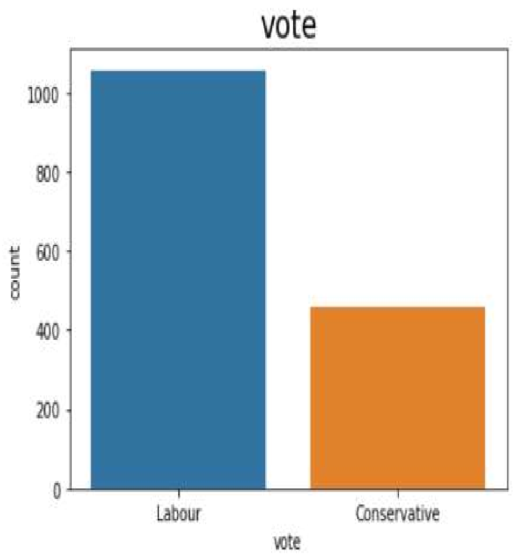
```
sns.barplot(data = df, x='vote',y='economic.cond.household')
```

```
<AxesSubplot:xlabel='vote', ylabel='economic.cond.household'>
```



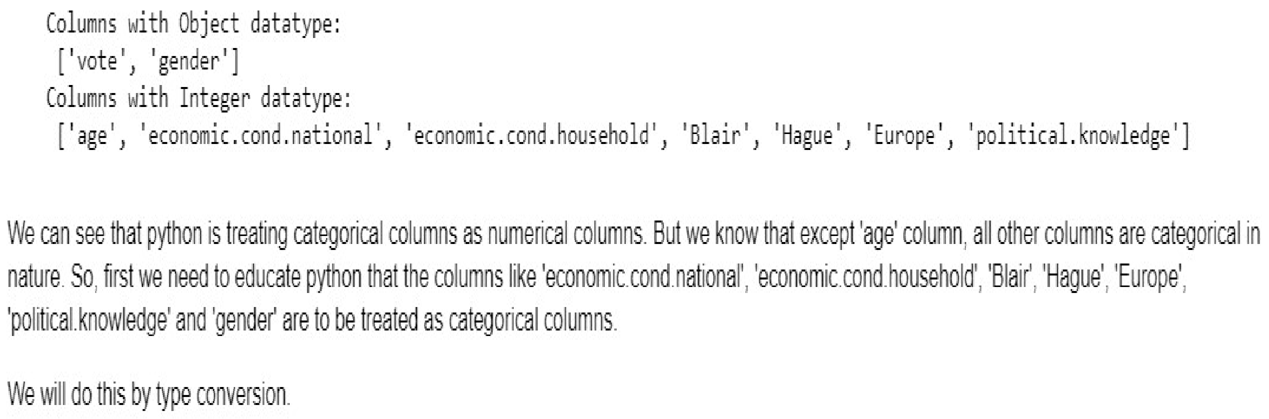


Count plot:



We can see from the above countplot that Labour party gets more than 1000+ votes, which is much better than the vote counts of the Conservative party.

Columns with different data types:

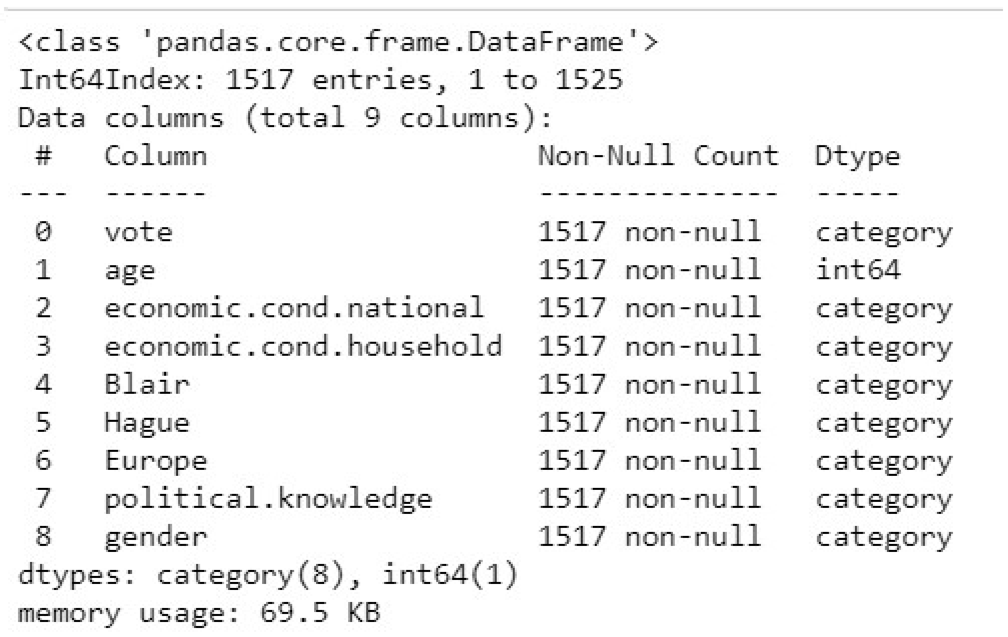


```
Columns with Object datatype:
 ['vote', 'gender']
Columns with Integer datatype:
 ['age', 'economic.cond.national', 'economic.cond.household', 'Blair', 'Hague', 'Europe', 'political.knowledge']
```

We can see that python is treating categorical columns as numerical columns. But we know that except 'age' column, all other columns are categorical in nature. So, first we need to educate python that the columns like 'economic.cond.national', 'economic.cond.household', 'Blair', 'Hague', 'Europe', 'political.knowledge' and 'gender' are to be treated as categorical columns.

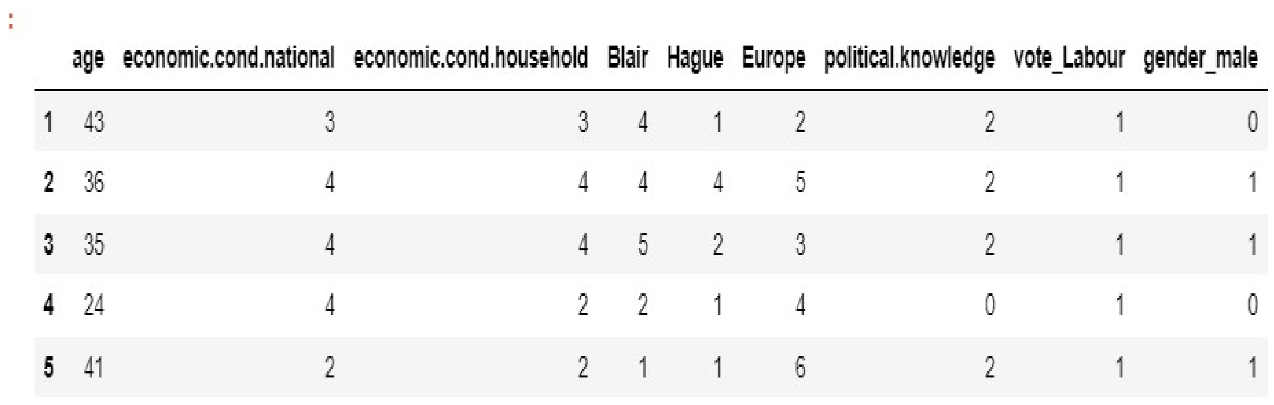
We will do this by type conversion.

After type conversion of the columns:



```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1517 entries, 1 to 1525
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   vote                     1517 non-null   category
 1   age                      1517 non-null   int64
 2   economic.cond.national   1517 non-null   category
 3   economic.cond.household  1517 non-null   category
 4   Blair                    1517 non-null   category
 5   Hague                    1517 non-null   category
 6   Europe                   1517 non-null   category
 7   political.knowledge      1517 non-null   category
 8   gender                   1517 non-null   category
dtypes: category(8), int64(1)
memory usage: 69.5 KB
```

Get dummies:



:

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	vote_Labour	gender_male
1	43	3	3	4	1	2	2	1	0
2	36	4	4	4	4	5	2	1	1
3	35	4	4	5	2	3	2	1	1
4	24	4	2	2	1	4	0	1	0
5	41	2	2	1	1	6	2	1	1

Scaling of the data:

Scaling of the data comes under the set of steps of data pre-processing when we are performing machine learning algorithms in the data set. Data standardization is the process of rescaling the attributes so that they have mean as 0 and variance as 1. The ultimate goal to perform standardization is to bring down all the features to a common scale without distorting the differences in the range of the values.

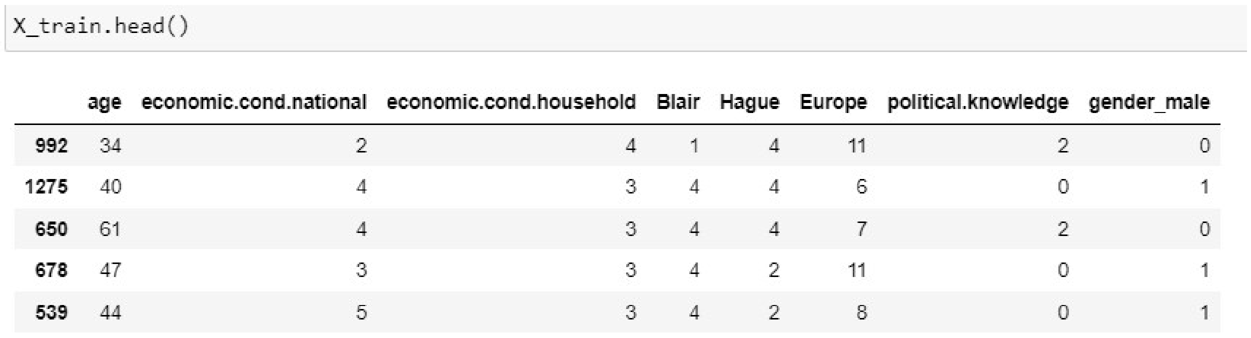
To bring all features in the same standing, we need to do scaling so that one significant number does not impact the model just because of their large magnitude.

Features scaling is necessary for this dataset, because we have seen in the boxplot that the 'age' feature has the numeric values which are much greater than the values present in the other features of the dataset.

Also we found out from the description (describe function) of the dataset that standard deviation of the 'age' is much higher than the standard deviation of the other available features in the dataset.

So, to normalize the range of independent variables or features of data, we will be scaling our data before any further analysis.

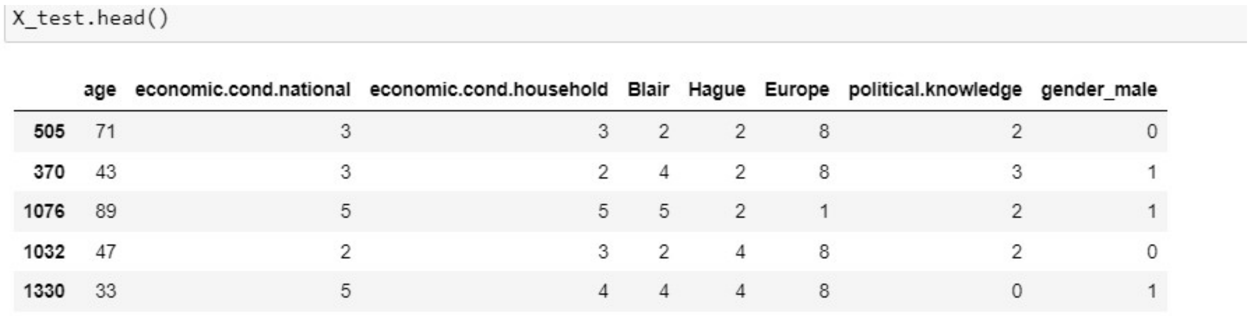
Train set:



```
X_train.head()
```

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
992	34	2	4	1	4	11	2	0
1275	40	4	3	4	4	6	0	1
650	61	4	3	4	4	7	2	0
678	47	3	3	4	2	11	0	1
539	44	5	3	4	2	8	0	1

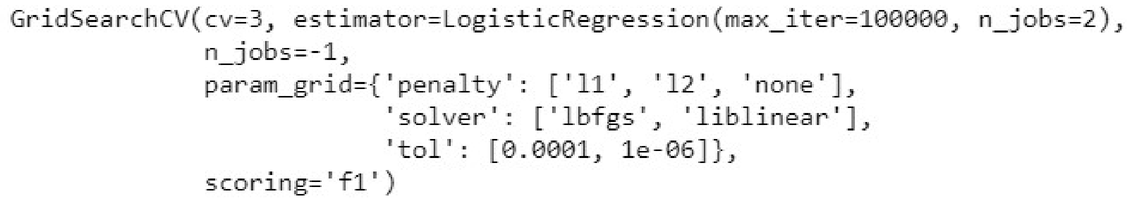
Test set:



```
X_test.head()
```

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
505	71	3	3	2	2	8	2	0
370	43	3	2	4	2	8	3	1
1076	89	5	5	5	2	1	2	1
1032	47	2	3	2	4	8	2	0
1330	33	5	4	4	4	8	0	1

Logistic regression:



```
GridSearchCV(cv=3, estimator=LogisticRegression(max_iter=100000, n_jobs=2),
             n_jobs=-1,
             param_grid={'penalty': ['l1', 'l2', 'none'],
                         'solver': ['lbfgs', 'liblinear'],
                         'tol': [0.0001, 1e-06]},
             scoring='f1')
```

We are using GridSearchCV to get the best parameters for the model building.

Using cv (cross validation) = 3, estimator is logistic regression with maximum iteration of 100000, n_jobs = 2.

Getting the best parameters , penalty:12, solver : ‘liblinear’, tol : 0.0001

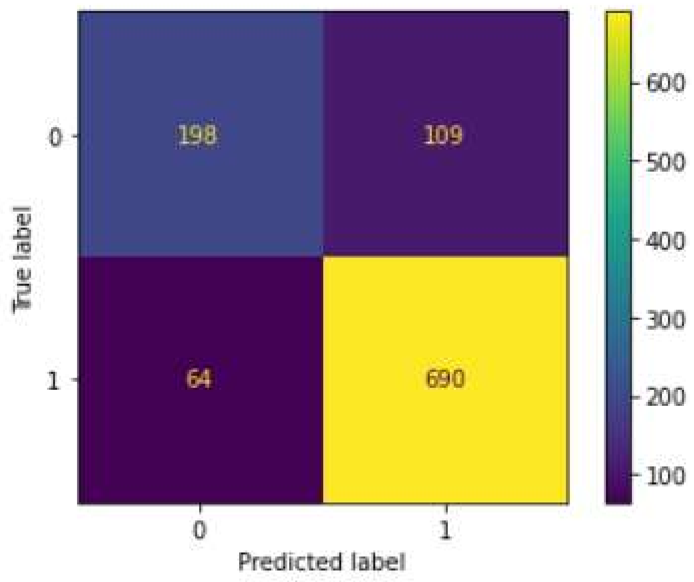
Predicting on Training and Test dataset and predicting the probabilities on test set.

Checking the head from the predicted probability.

	0	1
0	0.427675	0.572325
1	0.155830	0.844170
2	0.006244	0.993756
3	0.842250	0.157750
4	0.066400	0.933600
5	0.062857	0.937143
6	0.349056	0.650944
7	0.230190	0.769810
8	0.043898	0.956102
9	0.137002	0.862998

Confusion matrix:

For train data:



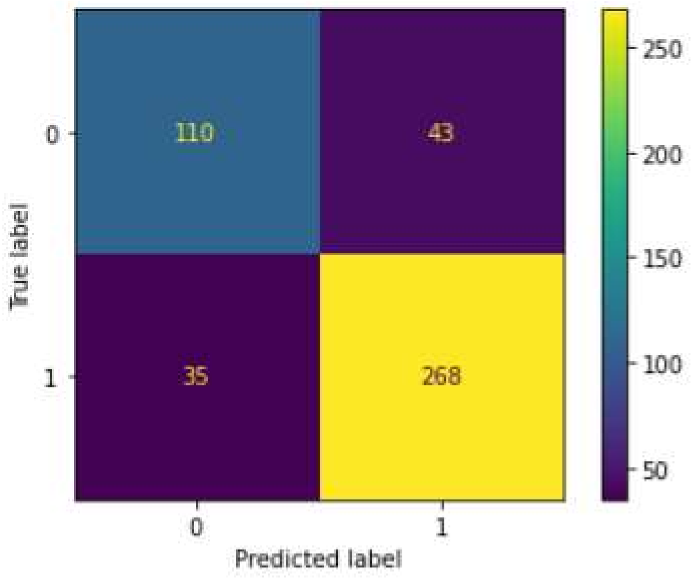
Classification Report:

```
print(classification_report(Y_train, ytrain_predict))
```

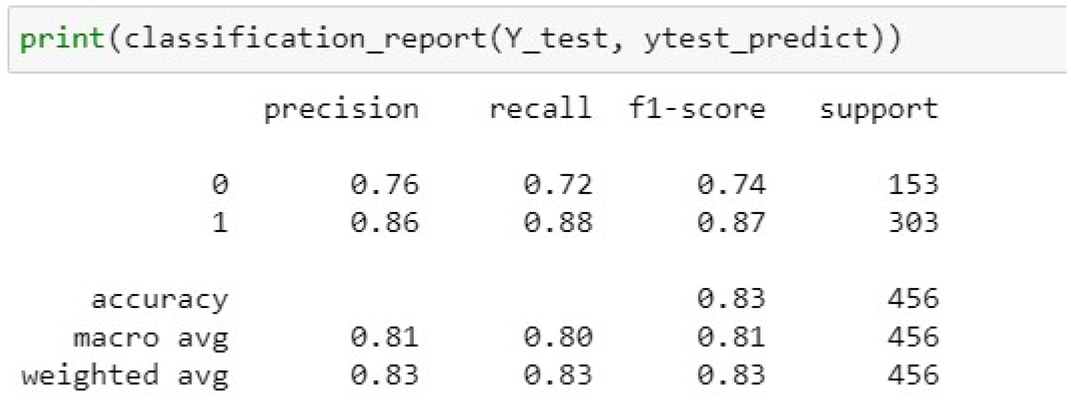
```
              precision    recall  f1-score   support

           0       0.76      0.64      0.70       307
           1       0.86      0.92      0.89       754

    accuracy                           0.84      1061
   macro avg       0.81      0.78      0.79      1061
weighted avg       0.83      0.84      0.83      1061
```

Confusion matrix:

For test data:



Classification Report:



```
print(classification_report(Y_test, ytest_predict))
```

```
              precision    recall  f1-score   support

           0       0.76      0.72      0.74       153
           1       0.86      0.88      0.87       303

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

Conclusion:

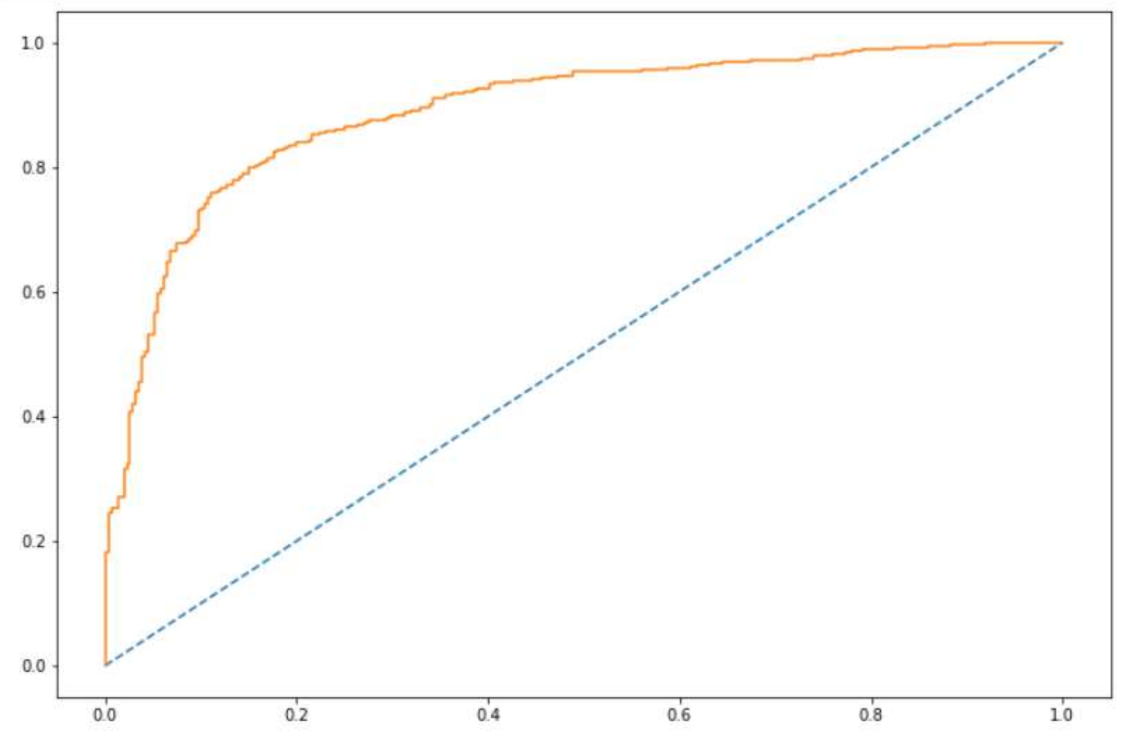
The accuracy of the model on train data is low than that of test data, that means the model is good to go.

The recall and f1 score also good in case of training data.

Area under curve and roc for the training data:

For train data:

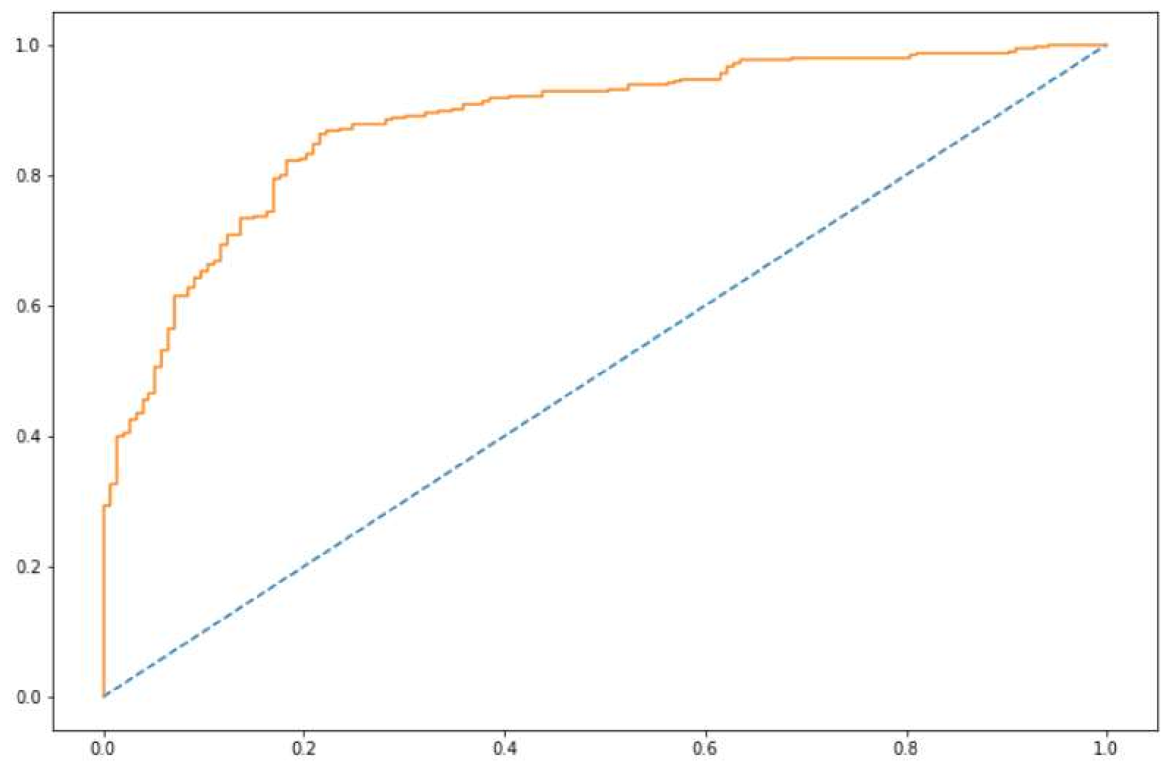
AUC: 0.890



For test:

Accuracy score: 0.8289473684210527

AUC: 0.880



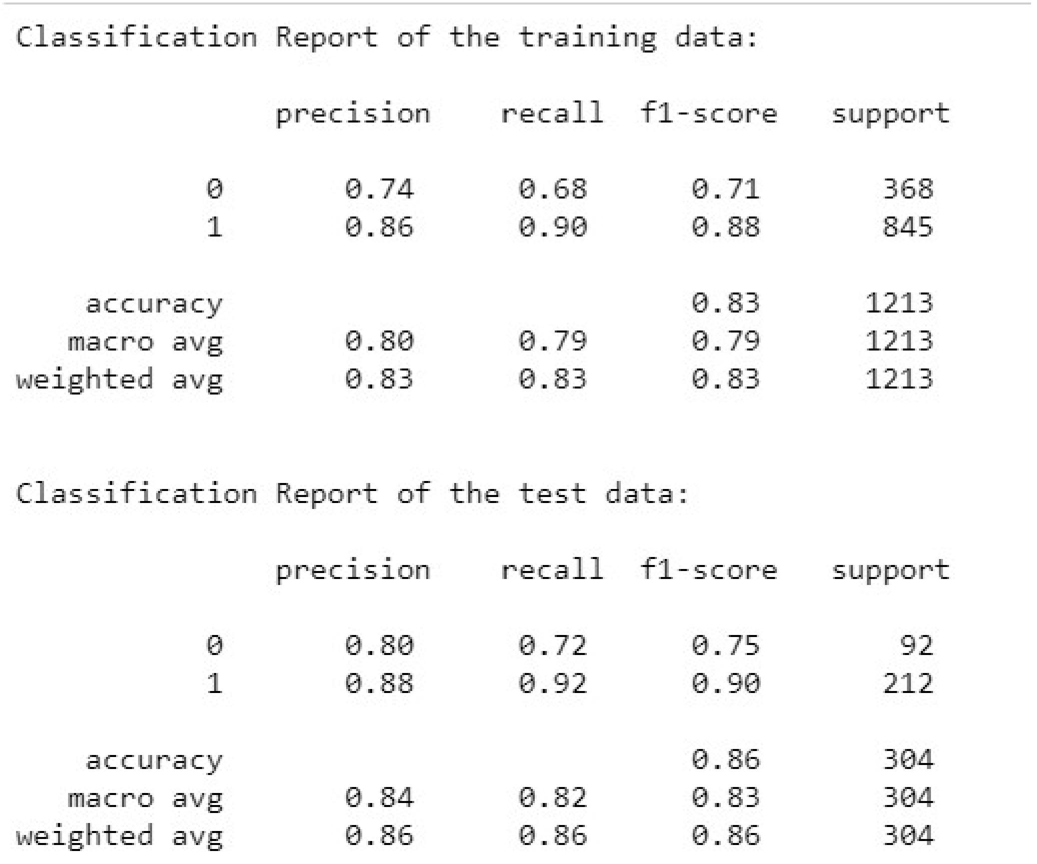
Linear Discriminant Analysis:

Imported LinearDiscriminantAnalysis from sklearn, and split the dataset into train and test set in 80:20 ratio.

Fitting the x_train and y_train into LDA classifier and then predicted the class for x_train and x_test.

Now, let see the classification report for the training and testing dataset:

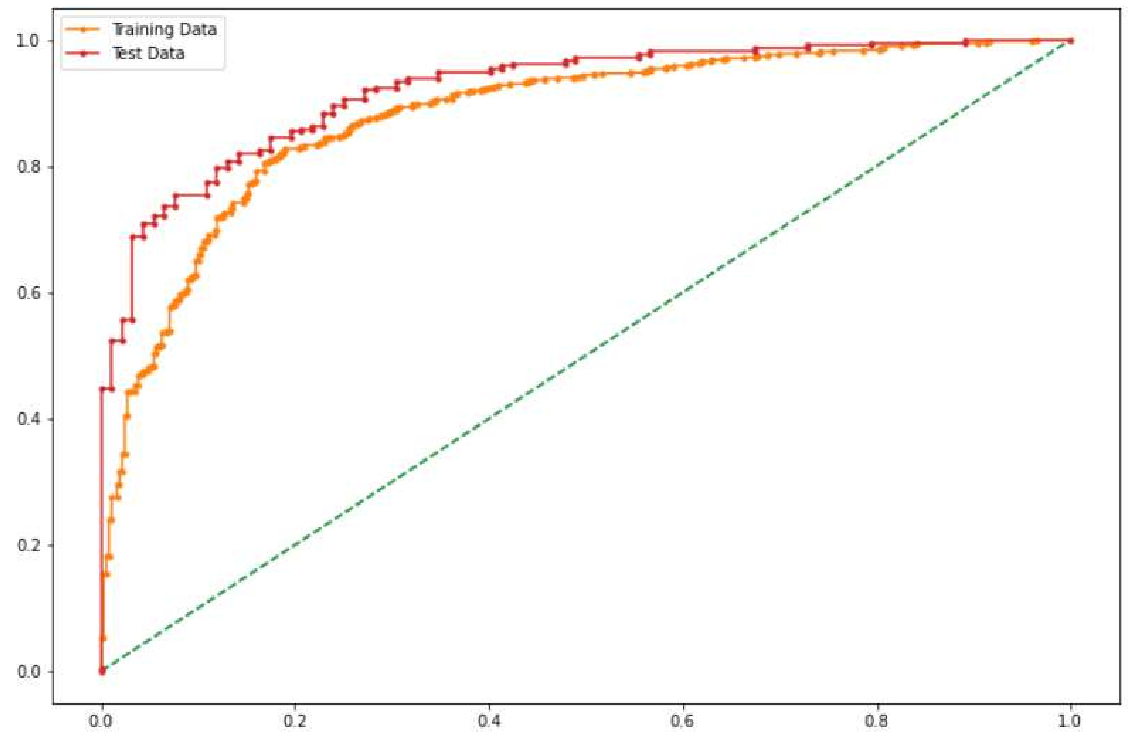
Classification Report for training and testing dataset:



```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       0.74      0.68      0.71       368
           1       0.86      0.90      0.88       845

    accuracy                           0.83      1213
   macro avg       0.80      0.79      0.79      1213
weighted avg       0.83      0.83      0.83      1213


Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.80      0.72      0.75        92
           1       0.88      0.92      0.90       212

    accuracy                           0.86       304
   macro avg       0.84      0.82      0.83       304
weighted avg       0.86      0.86      0.86       304
```

Let's check the Area under curve for training and test data set:

```
AUC for the Training Data: 0.880
AUC for the Test Data: 0.921
```



Conclusion:

The accuracy of the model for the train data is 83% and for the testing data is 86%.

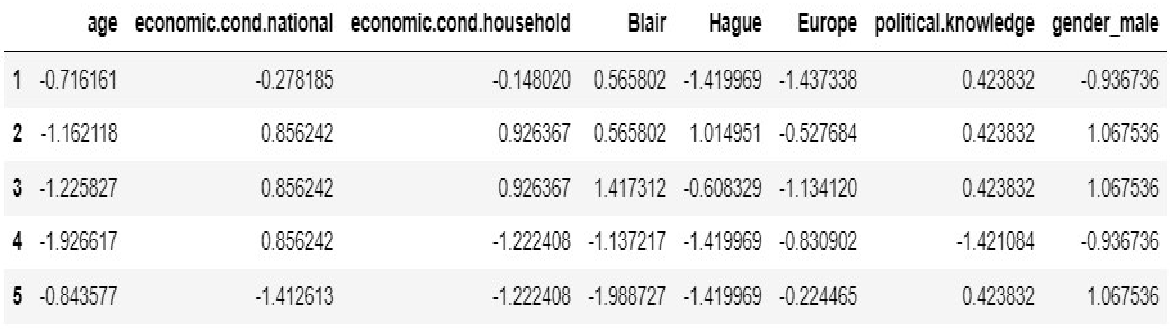
Recall and f1 score is also pretty good in testing data.

KNN model:

For KNN model, first we scale the data, because KNN model requires data to be scaled. Here we are using zscore to scale the data.

After separating the independent and dependent variables into X and Y, we will be scaling the numeric and categorical data values.

Checking the head of independent variables.



	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
1	-0.716161	-0.278185	-0.148020	0.565802	-1.419969	-1.437338	0.423832	-0.936736
2	-1.162118	0.856242	0.926367	0.565802	1.014951	-0.527684	0.423832	1.067536
3	-1.225827	0.856242	0.926367	1.417312	-0.608329	-1.134120	0.423832	1.067536
4	-1.926617	0.856242	-1.222408	-1.137217	-1.419969	-0.830902	-1.421084	-0.936736
5	-0.843577	-1.412613	-1.222408	-1.988727	-1.419969	-0.224465	0.423832	1.067536

Importing the KNeighbourClassifier from sklearn.neighbors then fitting the X_train and Y_train to the model.

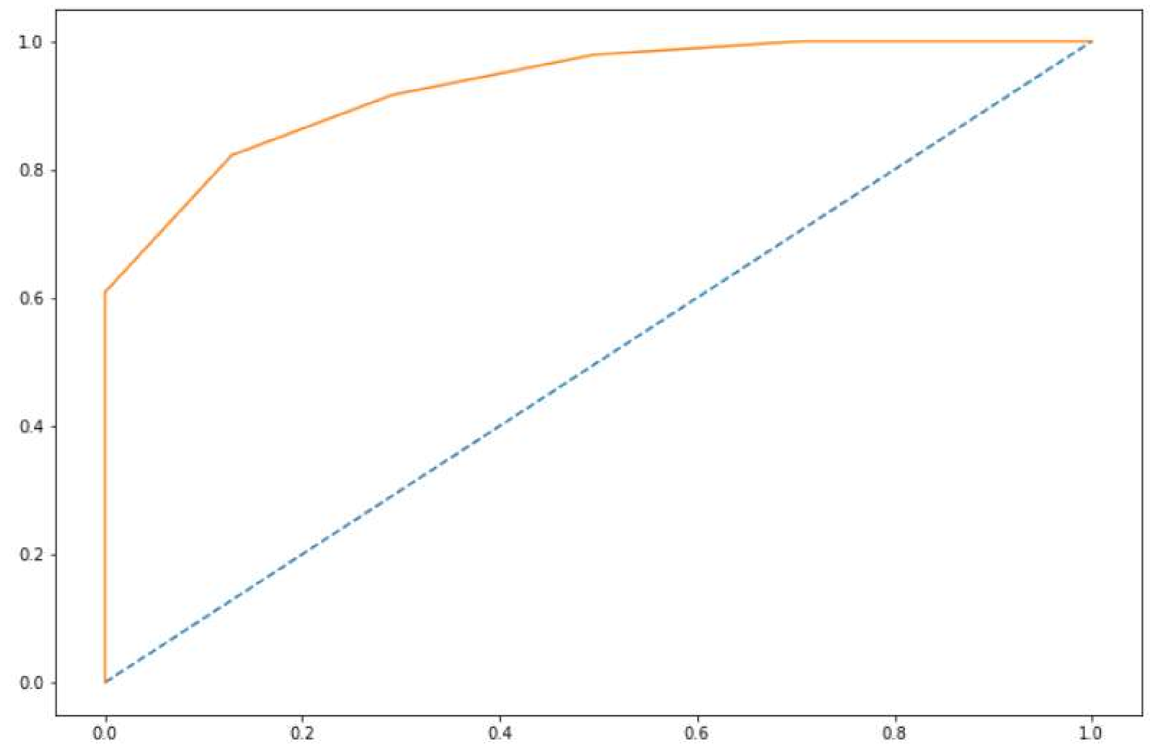
KNN model score: 0.85

Confusion Matrix and Classification Report:

```
Confusion Matrix

[[232  95]
 [ 68 742]]
Classification Report

              precision    recall  f1-score   support

           0       0.77      0.71      0.74       327
           1       0.89      0.92      0.90       810

    accuracy                           0.86      1137
   macro avg       0.83      0.81      0.82      1137
weighted avg       0.85      0.86      0.85      1137
```

Area Under Curve :

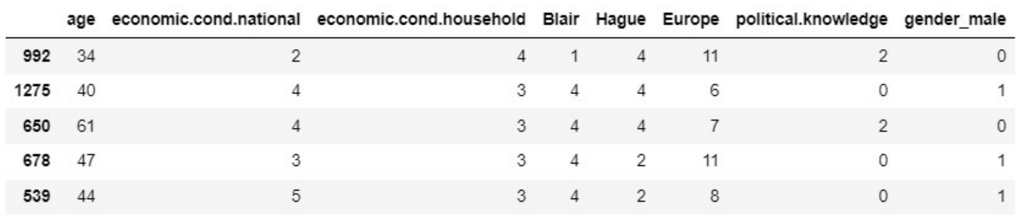
```
auc 0.929
```



Naïve Bayes model: Naïve Bayes does not need features to be scaled, that's why we won't scale our data.

So after separating the independent and dependent variables into X and Y and splitting into train and test set into 70:30 ratio.

Checking the head of X_train:



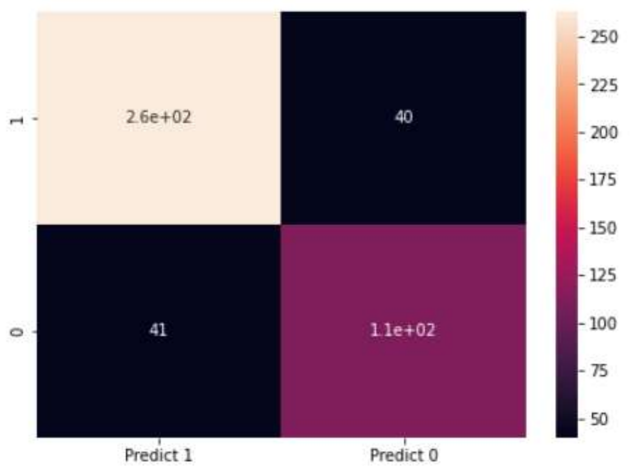
	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender_male
992	34	2	4	1	4	11	2	0
1275	40	4	3	4	4	6	0	1
650	61	4	3	4	4	7	2	0
678	47	3	3	4	2	11	0	1
539	44	5	3	4	2	8	0	1

After fitting the model into x_train and y_train we will see the performance of our model on train and test data.

Model accuracy on training data: 84%

Model accuracy on test data: 82%

Confusion Matrix and Classification Report:



Classification Report:

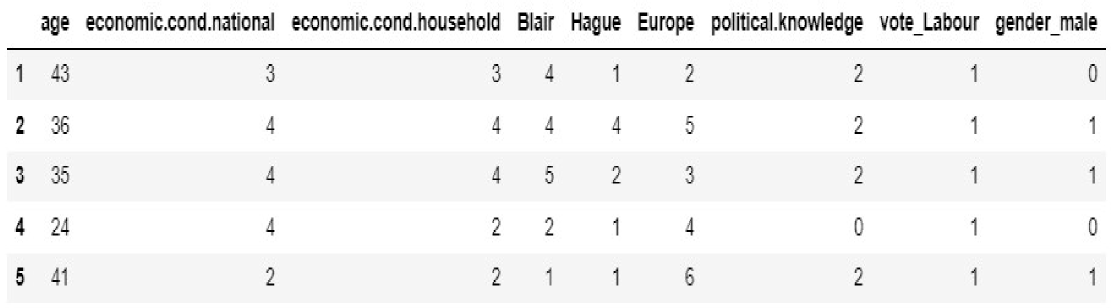
```
Classification Report
              precision    recall  f1-score   support

           1       0.87      0.87      0.87       303
           0       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
```

The precision for 1 is above 80% and accuracy is also above 80%

Model Tuning, Bagging (Random Forest will be applied for Bagging) and Boosting:

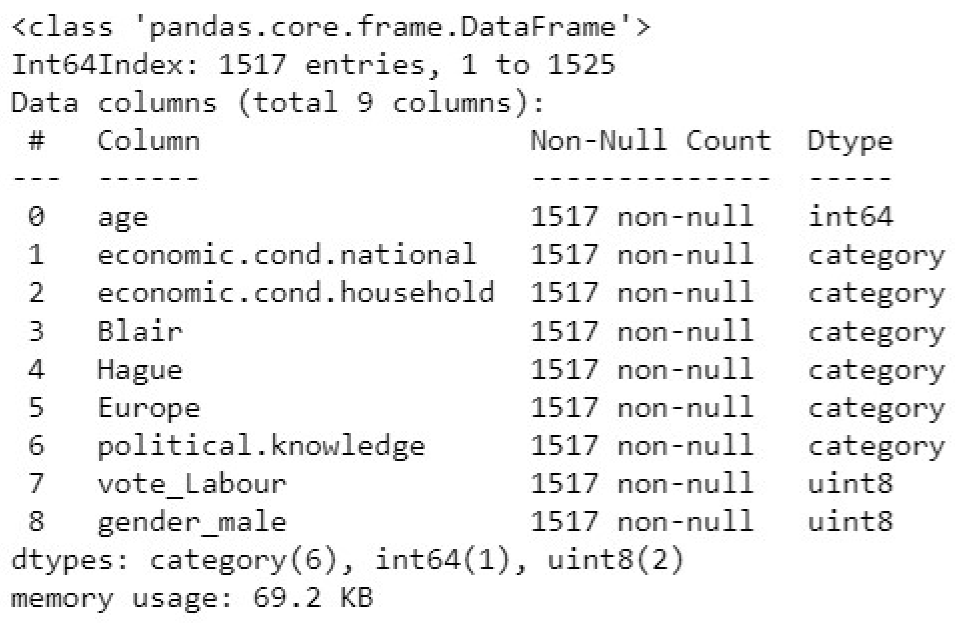
First we will importing RandomForestClassifier from sklearn.ensemble.

Checking the head of the data after get dummies.



	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	vote_Labour	gender_male
1	43	3	3	4	1	2	2	1	0
2	36	4	4	4	4	5	2	1	1
3	35	4	4	5	2	3	2	1	1
4	24	4	2	2	1	4	0	1	0
5	41	2	2	1	1	6	2	1	1

Info of the dataset:


```
<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   category
 2   economic.cond.household  1517 non-null   category
 3   Blair                    1517 non-null   category
 4   Hague                    1517 non-null   category
 5   Europe                   1517 non-null   category
 6   political.knowledge      1517 non-null   category
 7   vote_Labour              1517 non-null   uint8   
 8   gender_male              1517 non-null   uint8   
dtypes: category(6), int64(1), uint8(2)
memory usage: 69.2 KB
```

Note:

n_estimators is the number of trees, that we want to build within the Random Forest Classifier. Out of bag (oob) score is a way of validating the random forest model. OOB score is computed as the number of correctly predicted rows from the out of bag sample. Max_features will be used to select randomly features from the available number of features.

Rfcl.oob_score is 0.823751178133836

Error_rate(in %) : 0.17624882186616397

For the above specified grid parameters, we got the rfcl model with oob score 80.6% and error rate 19.3%, which shows the model is good to go for further evaluation.

Feature importance:

```
                              Imp
Hague                    0.406048
Blair                    0.302594
Europe                   0.205316
economic.cond.national   0.038313
political.knowledge      0.021143
age                      0.019404
economic.cond.household  0.005826
gender_male              0.001356
```

Hague and Blair are the features in the dataset with most importance.

Let's Now use Ensemble Learning methods and see the result

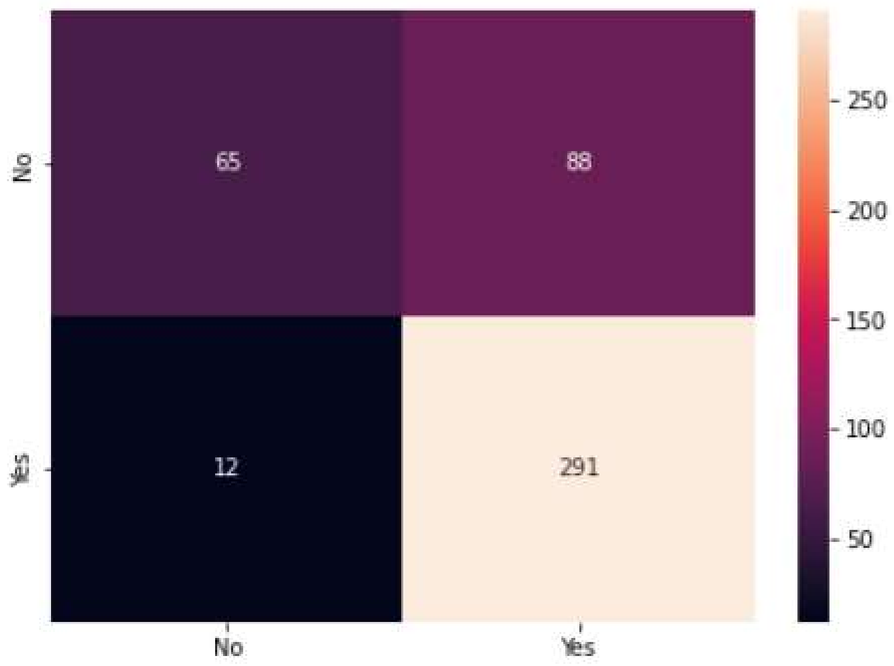
We will use Bagging , Ada Boosting and GradientBoost

Let's first look by use Bagging:

```python
#First import BaggingClassifier from sklearn.ensemble

#we will be using random forest as base_estimator

from sklearn.ensemble import BaggingClassifier

bgcl = BaggingClassifier(base_estimator=rfcl, n_estimators=50,random_state=1)
#bgcl = BaggingClassifier(n_estimators=50,random_state=1)

bgcl = bgcl.fit(X_train, y_train)
```

```python
y_predict = bgcl.predict(X_test)

print(bgcl.score(X_test , y_test))

cm=metrics.confusion_matrix(y_test, y_predict,labels=[0, 1])

df_cm = pd.DataFrame(cm, index = [i for i in ["No","Yes"]],
                  columns = [i for i in ["No","Yes"]])
plt.figure(figsize = (7,5))
sns.heatmap(df_cm, annot=True ,fmt='g')
```

```
0.7807017543859649
```

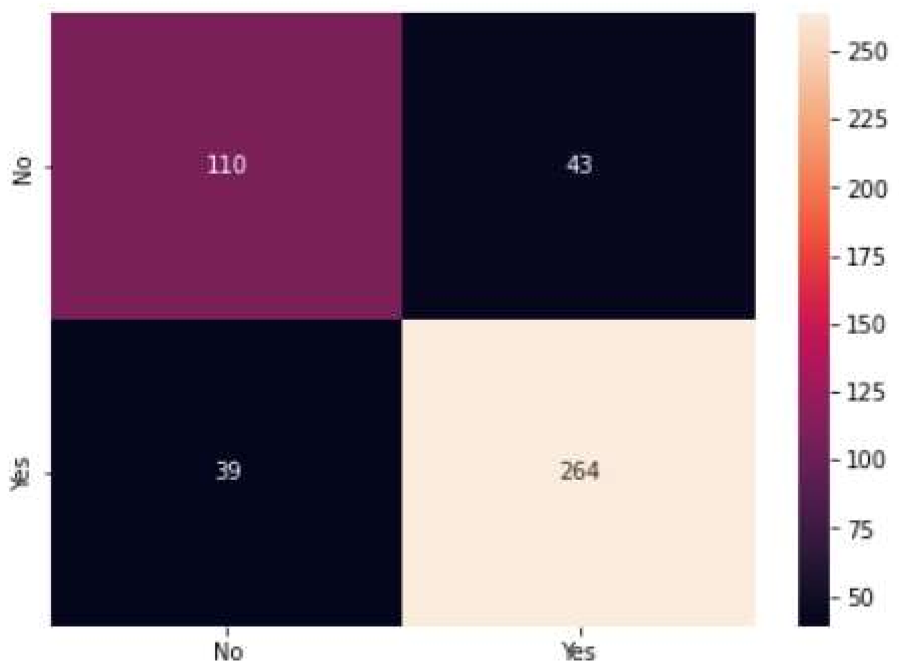


Ada Boosting

```
#first importing AdaBoostClassifier from sklearn.ensemble

from sklearn.ensemble import AdaBoostClassifier

abcl = AdaBoostClassifier(n_estimators=10, random_state=1)

#Applying .fit to the train data
abcl = abcl.fit(X_train, y_train)
```

```
y_predict = abcl.predict(X_test)
print(abcl.score(X_test , y_test))

cm=metrics.confusion_matrix(y_test, y_predict,labels=[0, 1])

df_cm = pd.DataFrame(cm, index = [i for i in ["No","Yes"]],
                  columns = [i for i in ["No","Yes"]])
plt.figure(figsize = (7,5))
sns.heatmap(df_cm, annot=True ,fmt='g')
```

```
0.8201754385964912
```



No
Yes
110
43
39
264
No
Yes
250
225
200
175
150
125
100
75
50

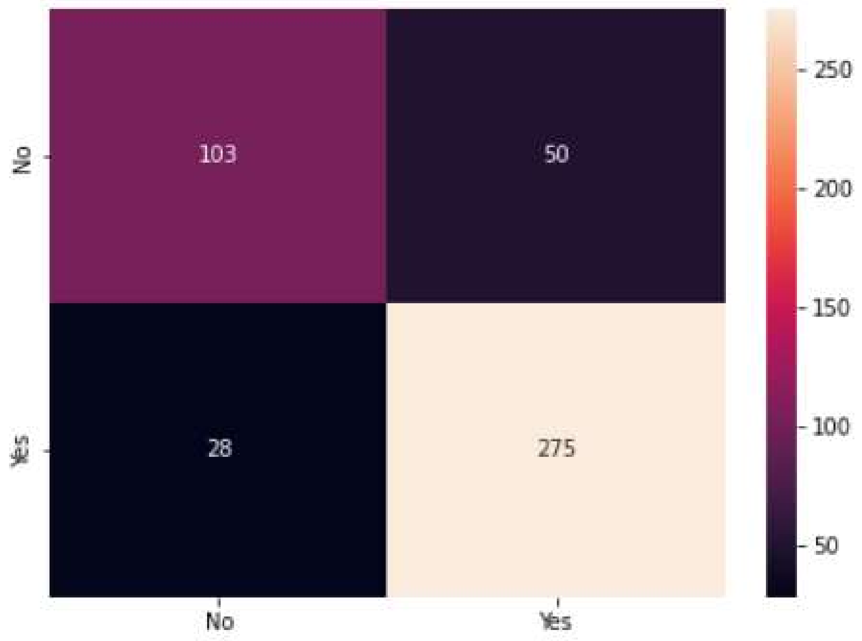
GradientBoost

```
from sklearn.ensemble import GradientBoostingClassifier
gbcl = GradientBoostingClassifier(n_estimators = 50,random_state=1)
gbcl = gbcl.fit(X_train, y_train)
```

```
y_predict = gbcl.predict(X_test)
print(gbcl.score(X_test, y_test))
cm=metrics.confusion_matrix(y_test, y_predict,labels=[0, 1])

df_cm = pd.DataFrame(cm, index = [i for i in ["No","Yes"]],
                  columns = [i for i in ["No","Yes"]])
plt.figure(figsize = (7,5))
sns.heatmap(df_cm, annot=True ,fmt='g')
```

```
0.8289473684210527
```

```
)4]: <AxesSubplot:>
```



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. (7 marks)

#performance metrics has already been performed on models

Logistic regression: for logistic regression on train data, we can see the below classification report:

The intention behind using logistic regression is to find the best fitting model to describe the relationship between the dependent and the independent variable.

The accuracy and recall value on the train data is above 80% which is a good indication for the labour party.

```
          precision    recall  f1-score   support


      0       0.76      0.64      0.70       307
      1       0.86      0.92      0.89       754


Accuracy                          0.84      1061
Macro avg 0.81 0.78 0.79 1061 weighted avg 0.83 0.84 0.83 1061
```

Same model when fit on test data:

```
          precision    recall  f1-score   support


      0       0.76      0.72      0.74       153
      1       0.86      0.88      0.87       303


Accuracy                          0.83       456
Macro avg 0.81 0.80 0.81 456 weighted avg 0.83 0.83 0.83 456
```

For train data, area under curve (auc) = 89%

And for test data , auc: 88%

For linear discriminant analysis

In linear discriminant analysis (lda):

Classification report of the training data:

```
          precision    recall  f1-score   support


      0       0.74      0.68      0.71       368
      1      0.86      0.90      0.88       845

Accuracy                          0.83      1213
```

Macro avg 0.80 0.79 0.79 1213 weighted avg 0.83 0.83 0.83 1213

Classification report of the test data:

```
          precision    recall  f1-score   support


      0       0.80      0.72      0.75        92
      1      0.88      0.92      0.90       212

Accuracy                          0.86       304
```

Macro avg 0.84 0.82 0.83 304 weighted avg 0.86 0.86 0.86 304

Auc for the training data: 88% auc for the test data: 92%

For knn model:

The model score is 85%

Classification report:

```
          precision    recall  f1-score   support


      0       0.77      0.71      0.74       327
      1      0.89      0.92      0.90       810
```

```
Accuracy                    0.86     1137
```

Macro avg 0.83 0.81 0.82 1137 weighted avg 0.85 0.86 0.85 1137

Auc on train data = auc 0.929

Naive bayes:

In train data model accuracy model accuracy: 0.84

On test data model accuracy: 0.82

Classification report:

```
        precision   recall  f1-score   support


    1     0.87     0.87     0.87      303
    0      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

The precision for 1 is above 80% and accuracy is also above 80%.

1.8 based on these predictions, what are the insights?

Insights:

By looking at the insights from the above predictions, we can confidently say that labour party would be the first choice to win in the elections. Below are some insights:

We can see that the labour party will get more number of votes that the conservative party.

Blair, who is the leader from labour party has received good ratings than The Hague from conservative party.

On a scale of 1 to 5, Blair has got rating 4 more than Hague in the assessment.

Female candidates are more active in voting than male.

Age does not have much impact on the result.

In all the models and their prediction on test data, the classification report seems to be in favor of 1(labour party votes)

On the basis of calculations based on current economic household condition and national economic condition, the labour party has got better performance representation than its opponent party.

MACHINE LEARNING

Problem 2:

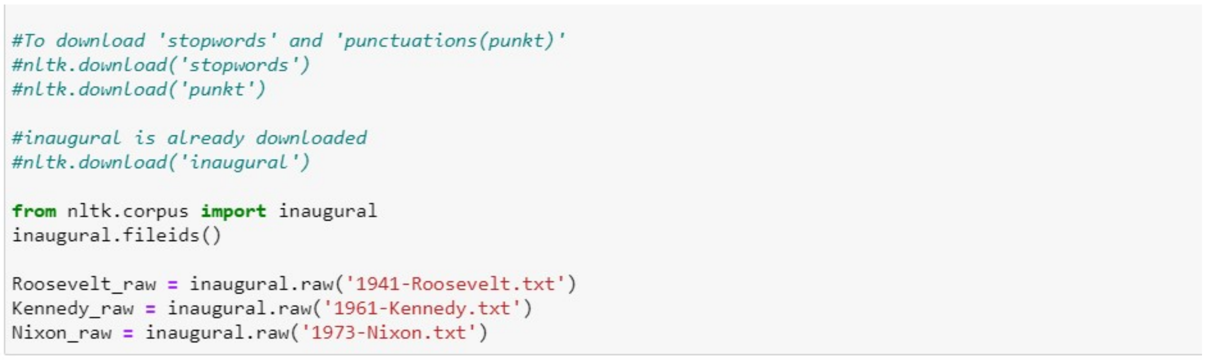
In this particular project, we are going to work on the inaugural corpora from the nltk in Python. We will be looking at the following speeches of the Presidents of the United States of America:

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

LOADING THE DATA:

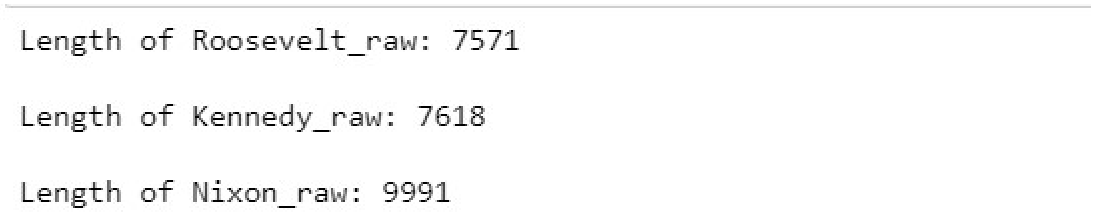
Importing all the required libraries and package.

Importing nltk (Natural Language Tool Kit) and string.



```
#To download 'stopwords' and 'punctuations(punkt)'
#nltk.download('stopwords')
#nltk.download('punkt')

#inaugural is already downloaded
#nltk.download('inaugural')

from nltk.corpus import inaugural
inaugural.fileids()

Roosevelt_raw = inaugural.raw('1941-Roosevelt.txt')
Kennedy_raw = inaugural.raw('1961-Kennedy.txt')
Nixon_raw = inaugural.raw('1973-Nixon.txt')
```

Now, let's check length of all 3 files:

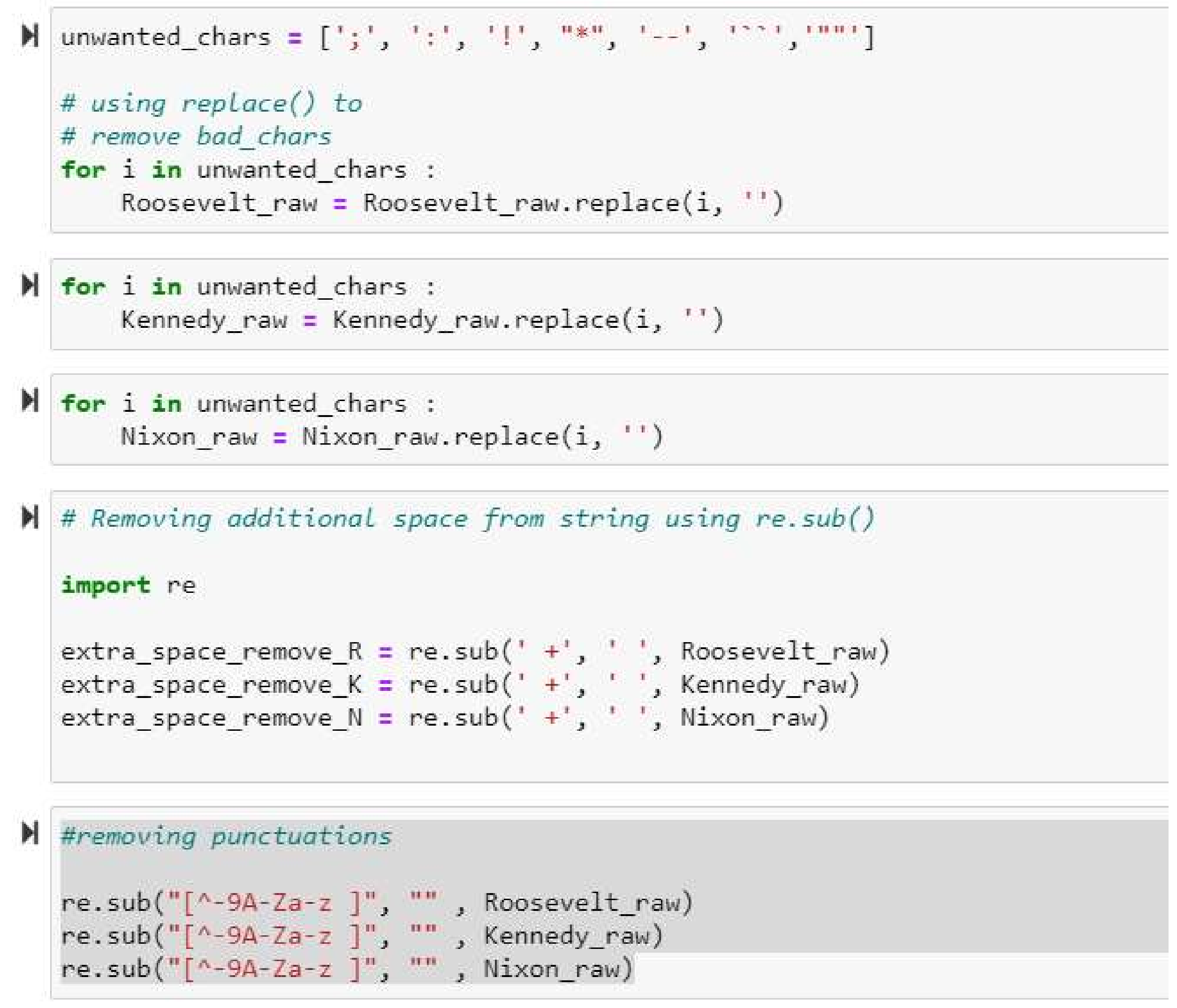


```
Length of Roosevelt_raw: 7571

Length of Kennedy_raw: 7618

Length of Nixon_raw: 9991
```

The variable 'Roosevelt_raw' contains a string with 7571 characters, Kennedy_raw contains 7618 characters, and Nixon_raw contains 9991 characters. This is the raw content of the speech, including many details we are not interested in such as whitespace, line breaks and blank lines. Notice the \r and \n in the opening line of the file, which is how Python displays the special carriage return and line feed characters. For our language processing, we want to break up the string into words and punctuation. This step is called tokenization, and it produces our familiar structure, a list of words and punctuation.

Removing unwanted characters:

Removing unwanted characters



```
unwanted_chars = [';', ':', '!', "*", '--', '``','""']

# using replace() to
# remove bad_chars
for i in unwanted_chars :
    Roosevelt_raw = Roosevelt_raw.replace(i, '')
```

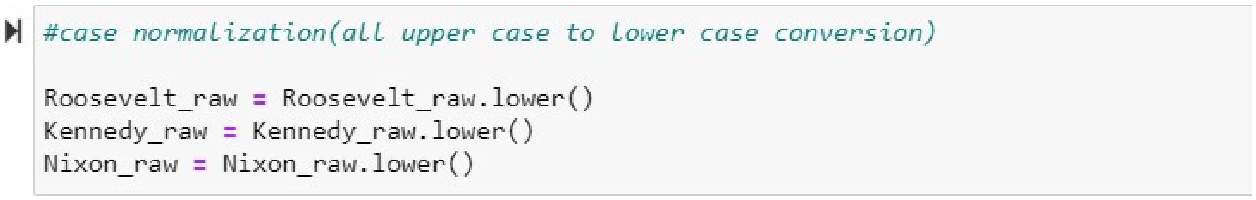
```
for i in unwanted_chars :
    Kennedy_raw = Kennedy_raw.replace(i, '')
```

```
for i in unwanted_chars :
    Nixon_raw = Nixon_raw.replace(i, '')
```

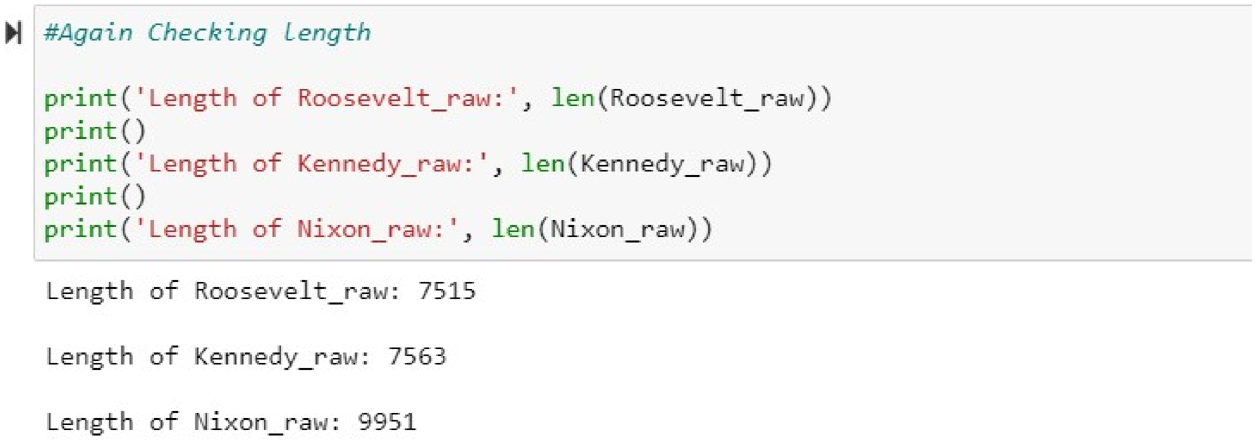
```
# Removing additional space from string using re.sub()

import re

extra_space_remove_R = re.sub(' +', ' ', Roosevelt_raw)
extra_space_remove_K = re.sub(' +', ' ', Kennedy_raw)
extra_space_remove_N = re.sub(' +', ' ', Nixon_raw)
```

```
#removing punctuations

re.sub("[^-9A-Za-z ]", "" , Roosevelt_raw)
re.sub("[^-9A-Za-z ]", "" , Kennedy_raw)
re.sub("[^-9A-Za-z ]", "" , Nixon_raw)
```

Converting all upper case words to lower case:



```
#case normalization(all upper case to lower case conversion)

Roosevelt_raw = Roosevelt_raw.lower()
Kennedy_raw = Kennedy_raw.lower()
Nixon_raw = Nixon_raw.lower()
```



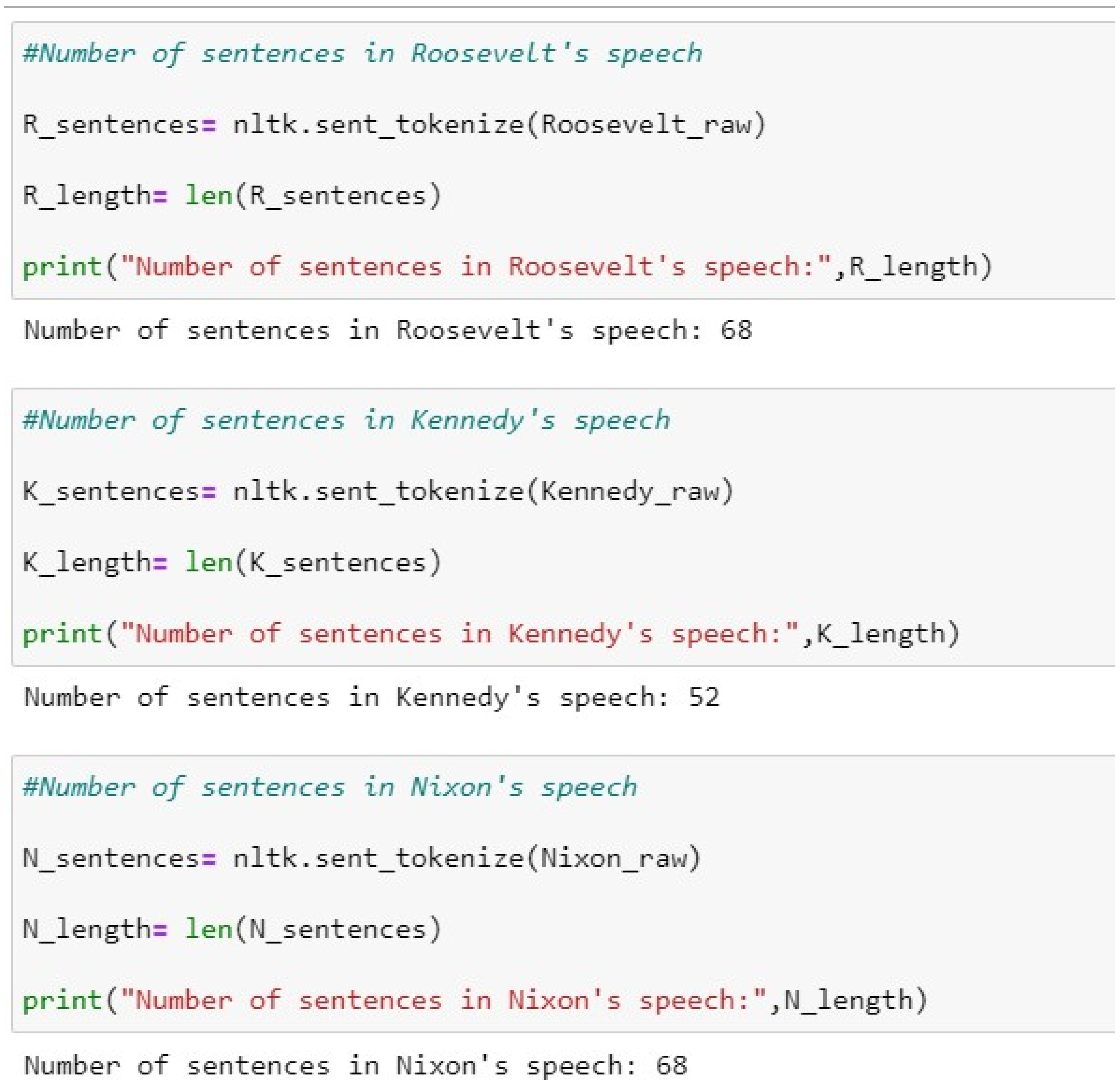
```
#Again Checking length

print('Length of Roosevelt_raw:', len(Roosevelt_raw))
print()
print('Length of Kennedy_raw:', len(Kennedy_raw))
print()
print('Length of Nixon_raw:', len(Nixon_raw))
```

```
Length of Roosevelt_raw: 7515

Length of Kennedy_raw: 7563

Length of Nixon_raw: 9951
```

Checking the length of the token after tokenization:

```
length of tokens: 1492
['on', 'each', 'national', 'day', 'of', 'inauguration', 'since', '1789', ',', 'the']
-----------------------------------------
length of tokens: 1512
['vice', 'president', 'johnson', ',', 'mr.', 'speaker', ',', 'mr.', 'chief', 'justice']
-----------------------------------------
length of tokens: 1983
['mr.', 'vice', 'president', ',', 'mr.', 'speaker', ',', 'mr.', 'chief', 'justice']
```

Checking Number of sentences in all 3 speeches:



```
#Number of sentences in Roosevelt's speech

R_sentences= nltk.sent_tokenize(Roosevelt_raw)

R_length= len(R_sentences)

print("Number of sentences in Roosevelt's speech:",R_length)
```

```
Number of sentences in Roosevelt's speech: 68
```

```
#Number of sentences in Kennedy's speech

K_sentences= nltk.sent_tokenize(Kennedy_raw)

K_length= len(K_sentences)

print("Number of sentences in Kennedy's speech:",K_length)
```

```
Number of sentences in Kennedy's speech: 52
```

```
#Number of sentences in Nixon's speech

N_sentences= nltk.sent_tokenize(Nixon_raw)

N_length= len(N_sentences)

print("Number of sentences in Nixon's speech:",N_length)
```

```
Number of sentences in Nixon's speech: 68
```

Checking texts from R_text:

```
['the', 'democratic', 'aspiration', 'is', 'no', 'mere', 'recent', 'phase', 'in', 'human',
'history', '.', 'it', 'is', 'human']
------------------------------
```

```
Checking texts from K_text:

 ['area', 'in', 'which', 'its', 'writ', 'may', 'run', '.', 'finally', ',', 'to', 'those', 'nations', '
who', 'would', 'make', 'themselves', 'our', 'adversary', ',']
------------------------------

Checking texts from N_text:

 ['to', 'new', 'has', 'not', 'been', 'a', 'retreat', 'from', 'our', 'responsibilities']

Checking the predefined set of stopwords:

Stopwords

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "
you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himsel
f', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'thei
r', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'th
ese', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had'
, 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', '
as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 't
hrough', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', '
out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'wh
en', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', '
some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't'
, 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're',
 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "does
n't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mig
htn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'should
n', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "
wouldn't"]

Again checking the length of tokens:

length of tokens: 1492
['on', 'each', 'national', 'day', 'of', 'inauguration', 'since', '1789', ',', 'the']
-----------------------------------
length of tokens: 1512
['vice', 'president', 'johnson', ',', 'mr.', 'speaker', ',', 'mr.', 'chief', 'justice']
```

```
--------------------------------------
length of tokens: 1983
['mr.', 'vice', 'president', ',', 'mr.', 'speaker', ',', 'mr.', 'chief', 'justice']
```

Checking the top 3 Most Frequent words:

For Roosevelt speech: Nation, know and Spirit.

```
[('nation', 12), ('know', 10), ('spirit', 9)]
```

For Kennedy: let, us and world.

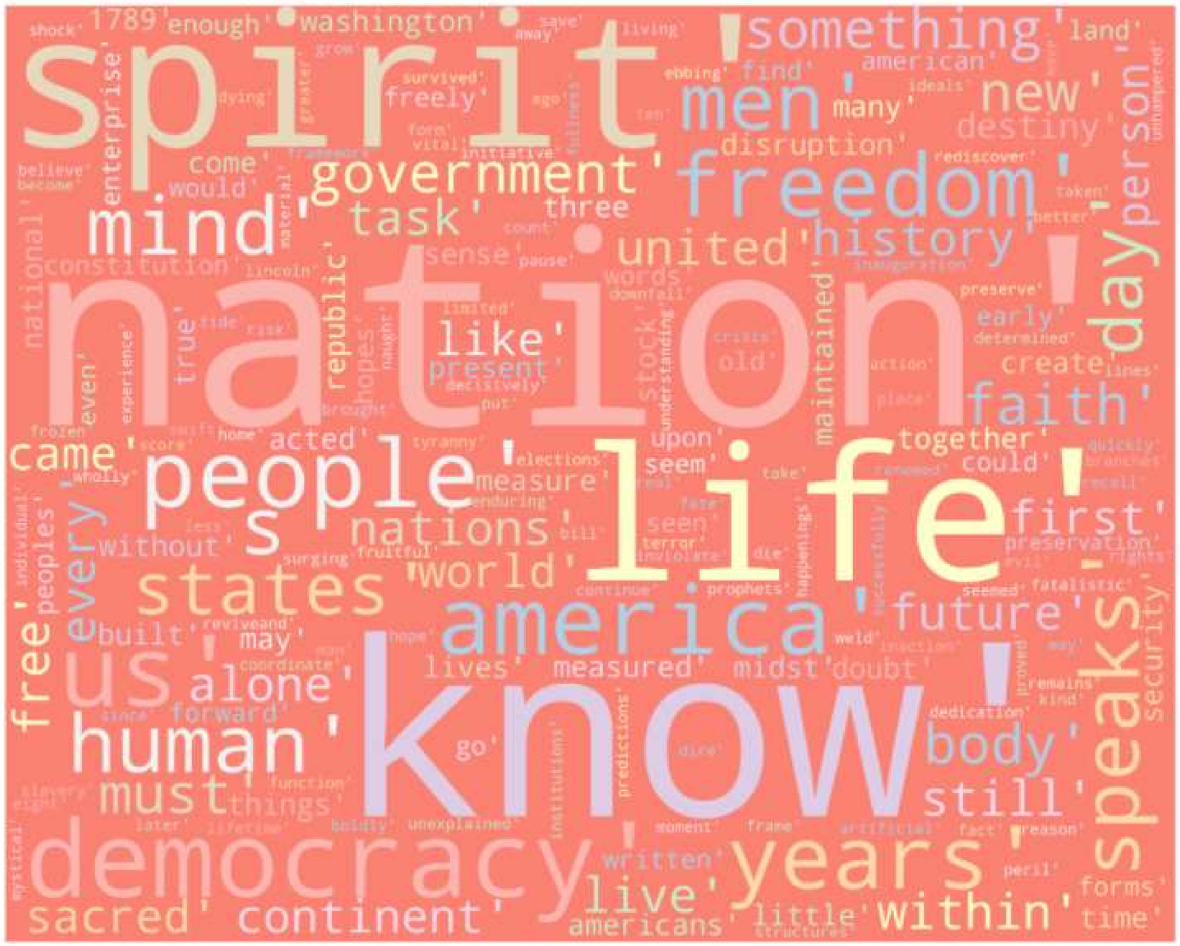
```
[('let', 16), ('us', 12), ('world', 8)]
```

For Nixon: us, let and America

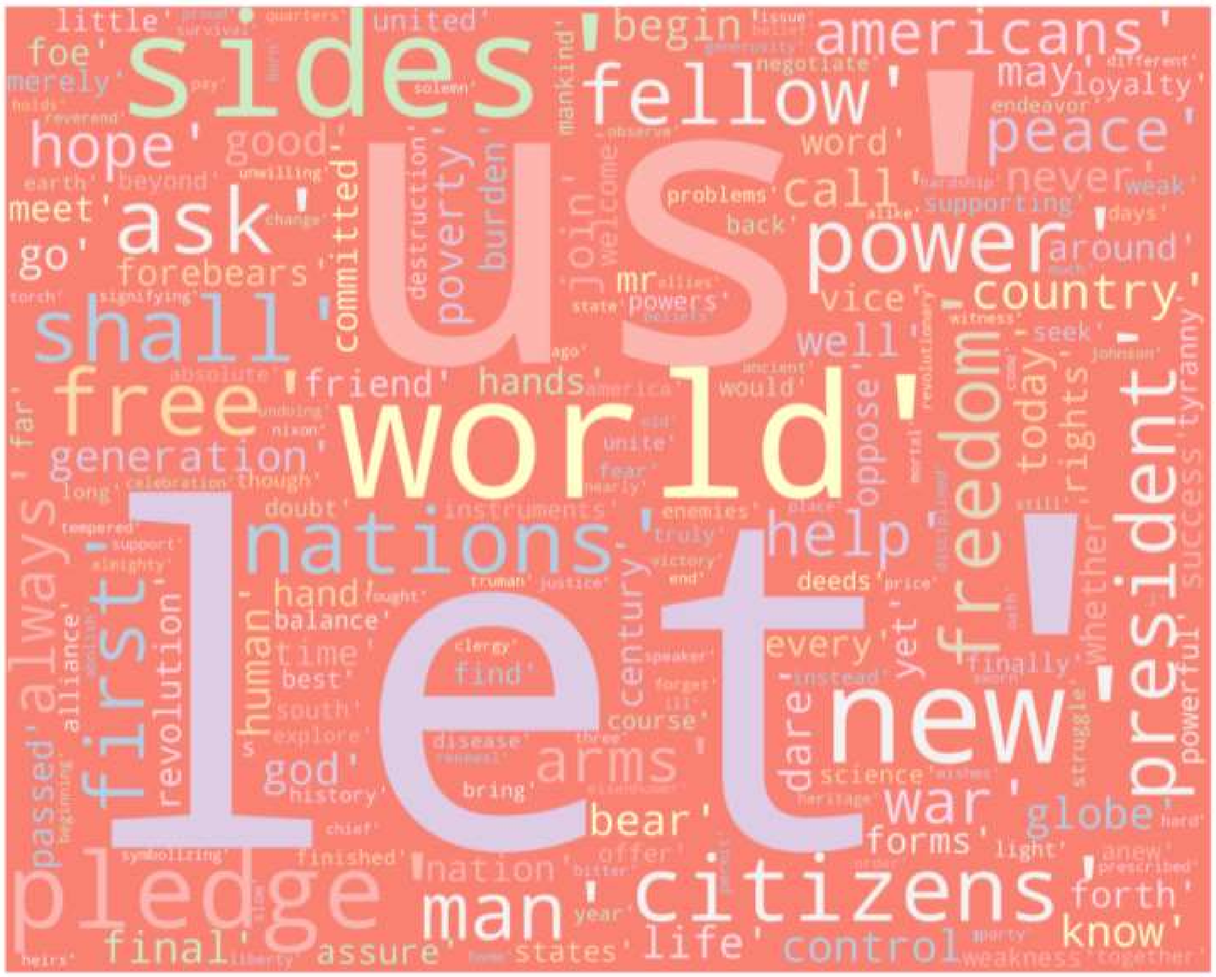
```
[('us', 26), ('let', 22), ('America', 21)]
```

Making the word cloud for all 3 speeches:

1. For Roosevelt:



2. For Kennedy:


us
let
world
sides
new
president
citizens
nations
pledge
freedom
power
shall
free
fellow
americans
ask
peace
man
first
hope
arms
war
help
country
generation
always
globe
call
begin
control

3. For Nixon:

us
let
america
peace
world
new
nation
responsibility
history
policies
government
great
together