MACHINE LEARNING BUSSINESS REPORT

PGPDSBA 2022 BATCH
BY AKASH JHA

CONTENTS:

PROJECT OBJECTIVE:

You are hired by one of the leading news channels CNBE who wants to analyse 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 Ingestion:

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

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

Data Preparation:

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).

Modelling:

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

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

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

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.

Inference:

1.8 Based on these predictions, what are the insights?

Problem 1 -Machine Learning:

Data Dictionary and Data Overview:

The Dataset Provided us stored 'Election_Data.xlsx' contains 1525 rows and 10 Variables namely.

Data Dictionary

1. **Vote**: Party choice: Conservative or Labour
2. A**ge**: in years
3. **Economic. Cond. national**: Assessment of current national economic conditions, 1 to 5.
4. E**conomic. 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.

Importing Libraries:

To import the dataset and perform Exploratory Data Analysis on the given dataset we imported the following packages:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import sklearn
from sklearn.model_selection import train_test_split
# for logistic regression
from sklearn.linear_model import LogisticRegression
# for Linear Discriminant Analysis
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

Solution 1.1:

Data Ingestion:

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

Read the data and do exploratory data analysis. Describe the data briefly. (Check the null values, Data types, shape, EDA, etc,).

Importing Data:

The dataset is imported in Jupyter Notebook by function **pd.read_csv() ,** Stored dataset in **"data_election" .**

Top Five Rows viewed by function **data_election.head().**

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

Shape of Dataset:

```
1  data_election.shape
```

```
(1525, 10)
```

Structure of Dataset:

Structure of dataset can be computed by using .info () function.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Unnamed: 0              1525 non-null   int64
 1   vote                    1525 non-null   object
 2   age                     1525 non-null   int64
 3   economic.cond.national  1525 non-null   int64
 4   economic.cond.household 1525 non-null   int64
 5   Blair                   1525 non-null   int64
 6   Hague                   1525 non-null   int64
 7   Europe                  1525 non-null   int64
 8   political.knowledge     1525 non-null   int64
 9   gender                  1525 non-null   object
dtypes: int64(8), object(2)
memory usage: 119.3+ KB
```

Check for duplicated & Null values:

There are no duplicated & Null Values in Dataset.

```
# check for null & Duplicated values
data_election.isna().sum()
# there are no null values.
```

```
Unnamed: 0                0
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
```

```
# check for duplicates
data_election.duplicated().sum()
# there are no duplicated values
```

```
0
```

Describe Dataset:

Dataset can be described by function by .describe() Function.

	Unnamed: 0	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	1525.000000
mean	763.000000	54.182295	3.245902	3.140328	3.334426	2.746885	6.728525	1.542295
std	440.373894	15.711209	0.880969	0.929951	1.174824	1.230703	3.297538	1.083315
min	1.000000	24.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000
25%	382.000000	41.000000	3.000000	3.000000	2.000000	2.000000	4.000000	0.000000
50%	763.000000	53.000000	3.000000	3.000000	4.000000	2.000000	6.000000	2.000000
75%	1144.000000	67.000000	4.000000	4.000000	4.000000	4.000000	10.000000	2.000000
max	1525.000000	93.000000	5.000000	5.000000	5.000000	5.000000	11.000000	3.000000

Inference:

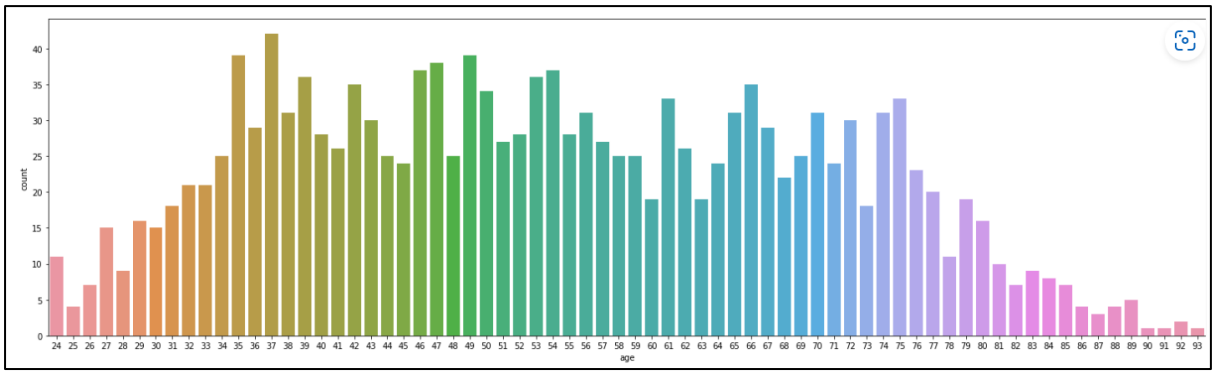
Observations from descriptive method

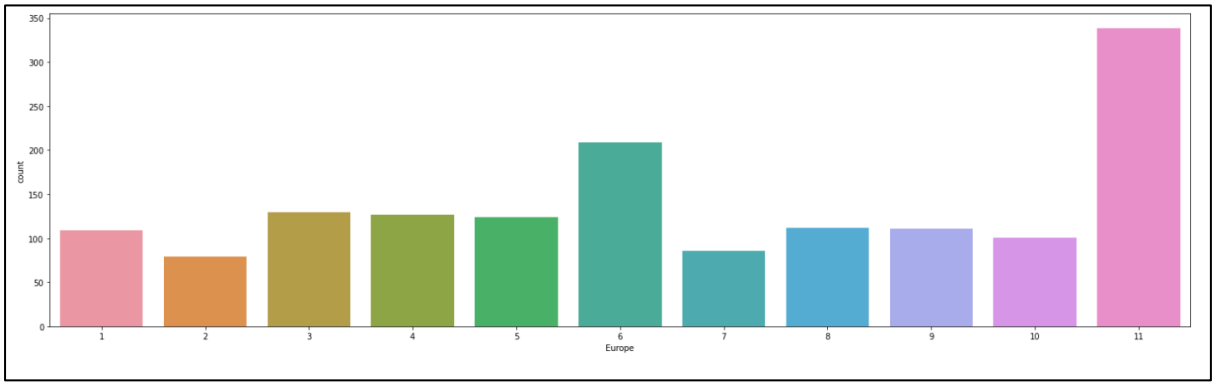
- Mean age for voters is 54 with 15 years deviation.
- Age variables have some outliers.
- political knowledge is of 25% voters is 0 and 50% voters are 2, which means most of voters are not more knowledgeable on stands of political party.
- Blair (labour leader) assessment is higher than Hague (Conservative leader)
- Economic Condition Assessment for household and Nation is mostly equal.

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

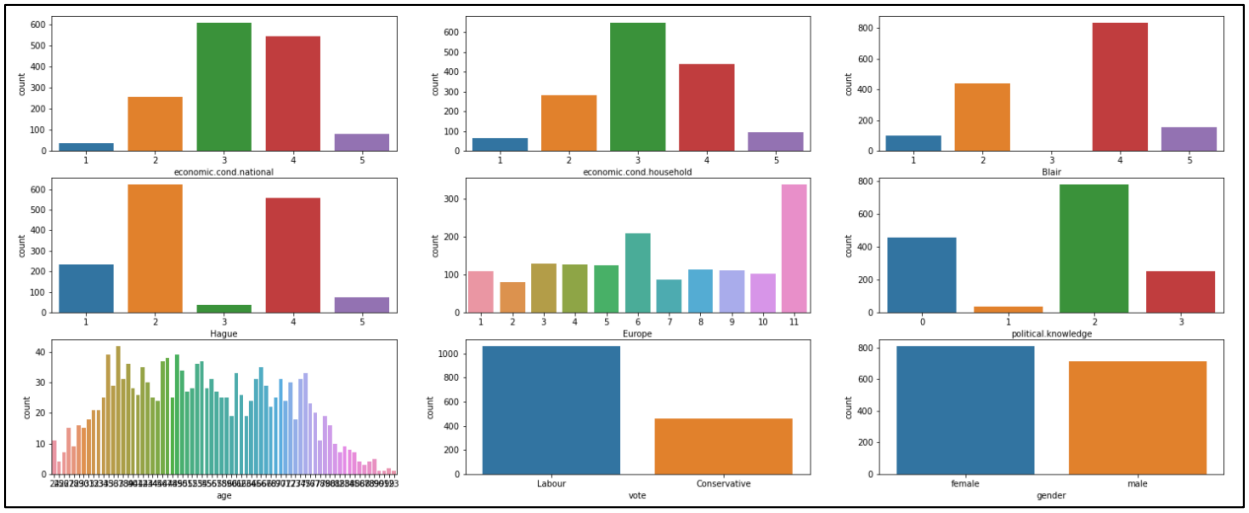
Univariate Analysis:

We plot count plot of all numerical variables by function sns.countplot() from seaborn package.





Data seems more voters have been Eurosceptic sentiment.

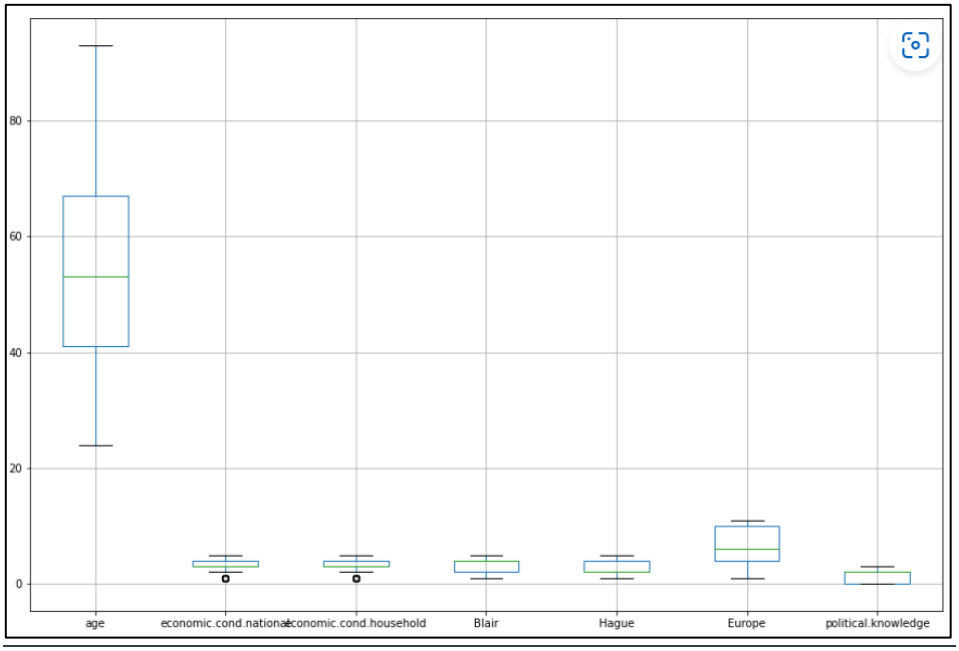


Inference:

- Major voting group population are age group of 30 to 80 yrs.
- There are more female voters than male.
- Major Group of people have low political Knowledge.

- Blair has more assessment points than Hague.
- Economic Condition of Household and National are Comparatively in same mid-range of voters.

Boxplot Before Scaling & Without treatment of Outliers:



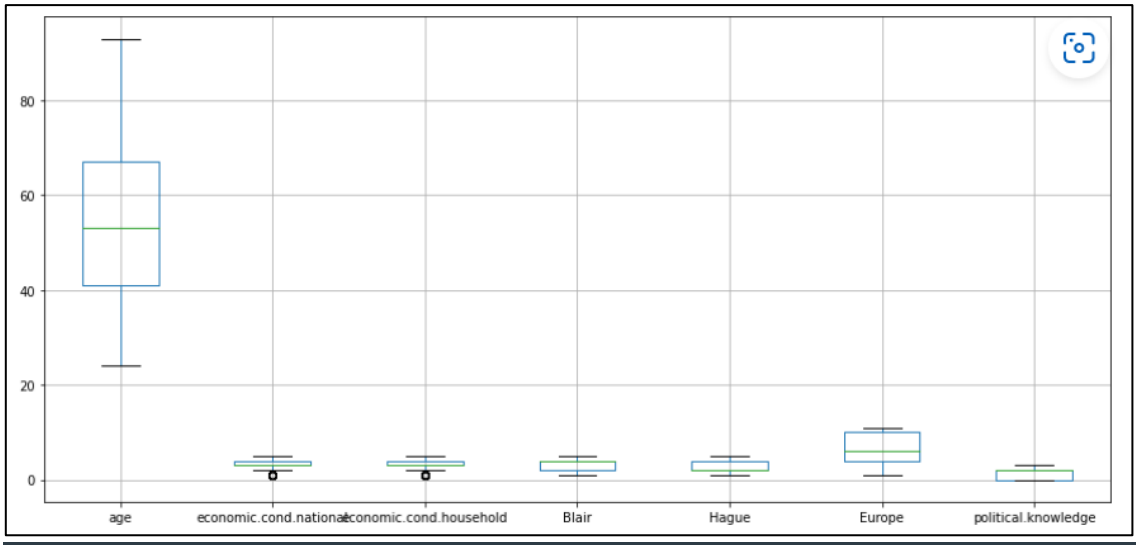
- **There is not many Outliers in any of Variables but still we can treat Outliers.**

Treating Outliers:

Treat Outliers by Apply method:

```
# there is no need to treat outliers
Q1= data_election.quantile(0.25)
Q3=data_election.quantile(0.75)
IQR=Q3-Q1
lr=(Q1-1.5*IQR)
ur=(Q3+1.5*IQR)

data_election_filtered=data_election[~((data_election<(lr)) | (data_election)>(ur)).any(axis=1)]
```

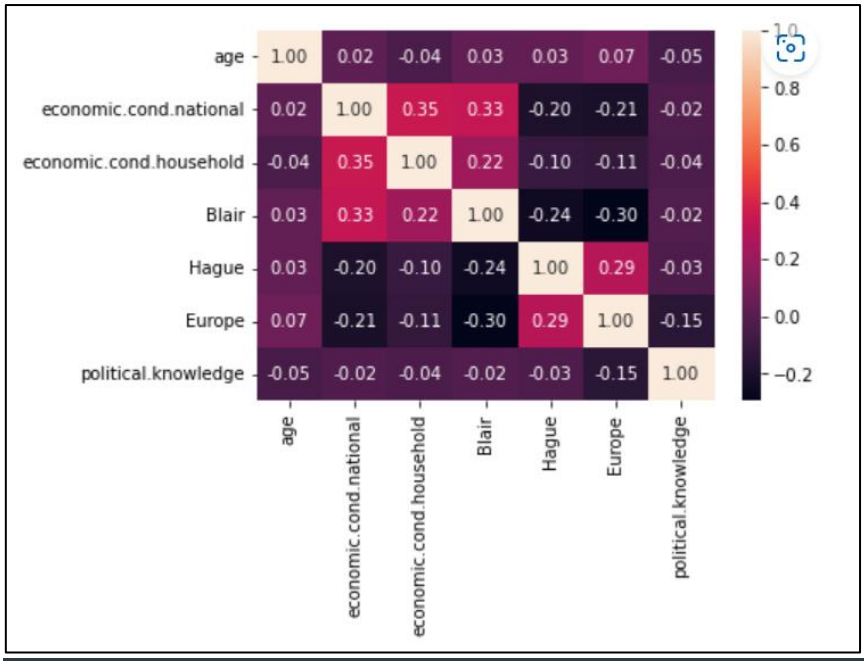
Boxplot after treating Outliers:



Multivariate Analysis:

Heat Map (Relationship Analysis)

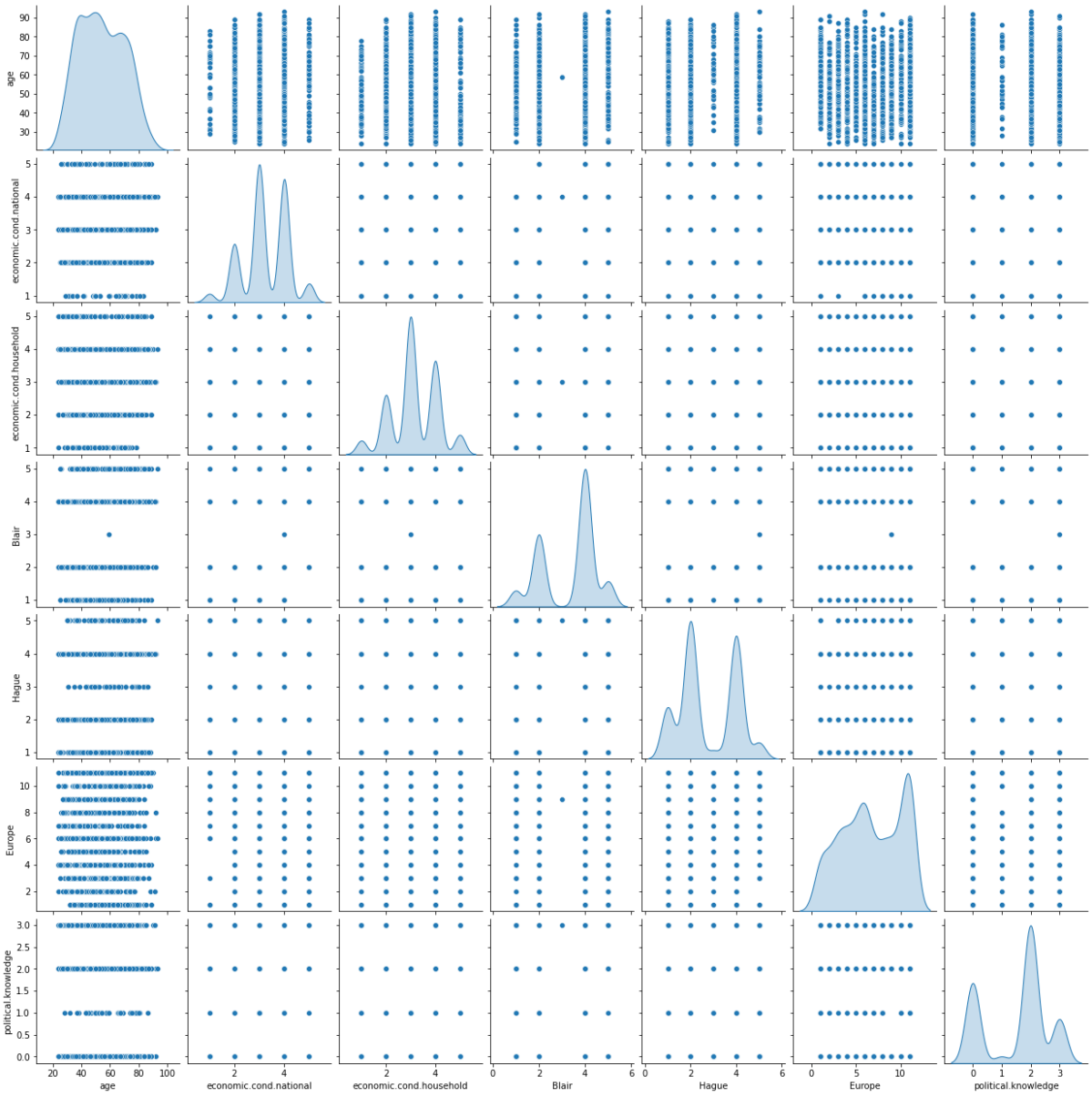
We will now plot a Heat Map or Correlation Matrix to evaluate the relationship between different variables in our dataset. This graph can help us to check for any correlations between different variables.



Inference:

- There is not much high correlation between variables.
- Still Economic. Condition. national is positively correlated with Economic condition household and Blair (Labour Leader)
- Europe is positively correlated with Hague (conservative Leader)
- i.e., voters who support for European union tends to favour for Conservative party Leader

Pair Plot for all the variables:



With the help of the above pair plot, we can understand the Univariate and Bivariate trends for all the variables in the dataset.

Inference:

- As you can observe no independent variable is in normal distribution shape.
- all variables have multiple peaks.
- Other than that, you cannot talk much about variable correlation for above scatter plots.

Data Preparation:

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 had encoded Variables with 0 & 1 of Votes & Gender.
- Changed Datatype of Votes and gender from object to category.

```
# Encode Data
data_election_filtered['vote']=np.where(data_election_filtered['vote']=='Labour',0,data_election_filtered['vote'])
data_election_filtered['vote']=np.where(data_election_filtered['vote']=='Conservative',1,data_election_filtered['vote'])
```

```
data_election_filtered['gender']=np.where(data_election_filtered['gender']=='female',0,data_election_filtered['gender'])
data_election_filtered['gender']=np.where(data_election_filtered['gender']=='male',1,data_election_filtered['gender'])
```

```
# Change data type of Objects to Category
data_election_filtered['vote']=data_election_filtered['vote'].astype('category')
data_election_filtered['gender']=data_election_filtered['gender'].astype('category')
```

- Scaling is Important for Logistic and LDA Models as they are regression model.
- But other classification Models doesn't get much affected by data is scaled or not.

As there are huge difference in value of Variables, we need to normalize data by Scaling for regression Models.

Splitting Data into 70:30 (70 % train and 30% test data):

- **Target Variable is Vote (Dependent)**

```
X=data_election_filtered.drop('vote',axis=1)
y=data_election_filtered['vote']
```

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

Dropping the target variable and making two data sets.

1.4) Apply Logistic Regression and LDA (Linear Discriminant Analysis) (2 pts). Interpret the inferences of both model s (2 pts). Successful implementation of each model. Logical reason behind the selection of different values for the parameters involved in each model. Calculate Train and Test Accuracies for each model. Comment on the validness of models (over fitting or under fitting)

Solution:

Before Apply LDA or Logistic Regression Model we need to normalize data as there is huge difference in variables and need to change the datatype of object to category.
But we will only scale the train data set.

for logistics and LDA we need to do Scaling

```
1 from sklearn.preprocessing import StandardScaler
2 scaler=StandardScaler()
```

```
1 scaler.fit(X_train)
```

```
7]: StandardScaler()
```

```
1 X_train1=scaler.transform(X_train)
2 X_test1=scaler.transform(X_test)
```

For Scaling we are using Standard Scalar Method and fitting the model in scaled data.

LDA MODEL

After Scaling we will fit scaled data in Linear Discriminant Analysis Model.

```
1 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
2 LDA=LinearDiscriminantAnalysis()
```

```
1 LDA.fit(X_train1,y_train)
```

```
LinearDiscriminantAnalysis()
```

```
1 print(LDA.score(X_train1,y_train))
```

```
0.8369259606373008
```

Accuracy for training Data set is 0.83.
Accuracy for Test Data set is 0.82

From Classification Report we can know recall & Precision of test Model:(As we can see this Data is Overfitted , at end we will analyaze with CV and Smote:

```
[[289  39]
 [ 44  86]]
              precision    recall  f1-score   support

           0       0.87      0.88      0.87       328
           1       0.69      0.66      0.67       130

    accuracy                           0.82       458
   macro avg       0.78      0.77      0.77       458
weighted avg       0.82      0.82      0.82       458
```

Logistic Regression Model:

Classification report and confusion matrix after fitting scaled data in Model show huge difference in train and test model. As Data is Overfitted for one Variable of target column the number of data is more and for other it's less. We can say data is skewed.

```
1 from sklearn.linear_model import LogisticRegression
```

```
1 log_res=LogisticRegression()
```

```
1 log_res.fit(X_train1,y_train)
```

```
LogisticRegression()
```

```
0.8397375820056232
0.8231441048034934
[[292  45]
 [ 36  85]]
              precision    recall  f1-score   support

           0       0.89      0.87      0.88       337
           1       0.65      0.70      0.68       121

    accuracy                           0.82       458
   macro avg       0.77      0.78      0.78       458
weighted avg       0.83      0.82      0.83       458
```

Train Accuracy-0.83
Test Accuracy -0.82
Precision-0.65
Recall- 0.70'

But we cannot say this Models are ideal as data is Overfitted and train model is too good compare to test model, Need to try other classification Model.

1.5) Apply KNN Model and Naïve Bayes Model (2pts). Interpret the inferences of each model (2 pts). Successful implementation of each model. Logical reason behind the selection of different values for the parameters involved in each model. Calculate Train and Test Accuracies for each model. Comment on the validness of models (over fitting or under fitting)?

- for KNN and Naive Bayes we don't need to scale data (As Classification is independent from effects of Outliers and extreme values)
- Import important libraries.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report,confusion_matrix
```

```
# Apply KNN model
NNH=KNeighborsClassifier(n_neighbors=5,weights='distance')
```

```
NNH.fit(X_train,y_train)
print(NNH.score(X_train,y_train))
print(NNH.score(X_test,y_test))
```

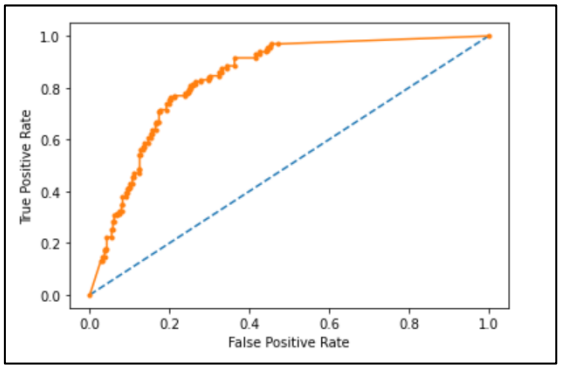
```
0.9990627928772259
0.7816593886462883
```

In Hyperparameters we are taking no of neighbours as 5 and distance as weight, we decide this by Model tuning.

Fit KNN in train and test Model, Accuracy for train model around 1 and test 0.78 , Definitely this model is not good.'

```
[[279  49]
 [ 51  79]]
              precision    recall  f1-score   support

           0       0.85      0.85      0.85       328
           1       0.62      0.61      0.61       130

    accuracy                           0.78       458
   macro avg       0.73      0.73      0.73       458
weighted avg       0.78      0.78      0.78       458
```

AUC Score for Train Data Set KNN Model: 1 & for test Data: 0.83



ROC Curve for Test Data KNN , It no where seems Good Model , when we compare for test & train.

Naive Bayes Model:

- For Naive Bayes we don't need to scale data (As Classification is independent from effects of Outliers and extreme values)
- Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. The Naive Bayes classifier works on the principle of conditional probability, as given by the Bayes theorem.

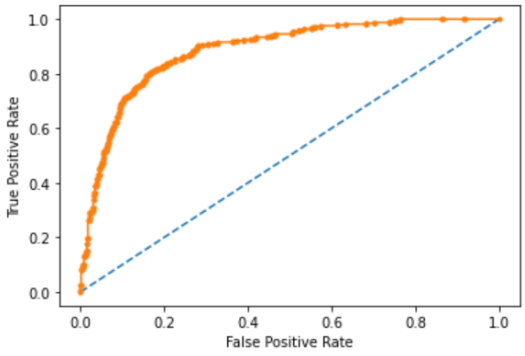
We will fit train and test data set in Naïve Bayes Model , as we are trying to predict election data and this model calculates on Probabilities, this model results must be good as per given criteria in question.

Fit Model in test and train, In Hyper parameter we are Using Gaussian () method to calculate Conditional probabilities. P(A|B) = P(B|A) * P(A) / P(B)

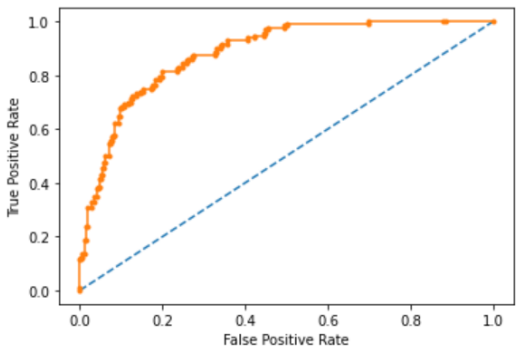
```
model=GaussianNB()
model.fit(X_train,y_train)
print(model.score(X_train,y_train))
print(model.score(X_test,y_test))

y_predict1=model.predict(X_test)
print(confusion_matrix(y_test,y_predict1))
print(classification_report(y_test,y_predict1))
```

```
0.8331771321462043
0.8253275109170306
[[284  44]
 [ 36  94]]
              precision    recall  f1-score   support

           0       0.89      0.87      0.88       328
           1       0.68      0.72      0.70       130

    accuracy                           0.83       458
   macro avg       0.78      0.79      0.79       458
weighted avg       0.83      0.83      0.83       458
```

Accuracy for Test and Train: 0.82 & 0.83
Recall & Precision Score are also liable for this model.
AUC Score Train: 0.88
AUC Score Test: 0.88



Train Data ROC Curve Naïve Bayes



Test Data ROC Curve Naïve Bayes

- By Naive Bayes method, we are getting good model , overfitting is not something we should worry that much with naive Bayes , Naive Bayes algorithm has tendency to underfit data

As Data is Overfitted we can fit model by SMOTE and analyaze results :

Firstly, import and apply Smote from imblearn. Oversampling library.

```
from imblearn.over_sampling import SMOTE
```

```
sm=SMOTE(random_state=1)
```

```
# Oversampling smote is only applied on train dataset
X_train_res,Y_train_res=sm.fit_resample(X_train,y_train.ravel())
```

```
Y_train_res.shape
```

```
(1470,)
```

By applying Smote method on KNN model we can see there is slight improvement in train model, but still it much higher than testing model.

```
0.8836734693877552
[[605 130]
 [ 41 694]]
              precision    recall  f1-score   support

           0       0.94      0.82      0.88       735
           1       0.84      0.94      0.89       735

    accuracy                           0.88      1470
   macro avg       0.89      0.88      0.88      1470
weighted avg       0.89      0.88      0.88      1470
```

Train Classification Report After Smote KNN Model

```
0.7554585152838428
[[246  82]
 [ 30 100]]
              precision    recall  f1-score   support

           0       0.89      0.75      0.81       328
           1       0.55      0.77      0.64       130

    accuracy                           0.76       458
   macro avg       0.72      0.76      0.73       458
weighted avg       0.79      0.76      0.77       458
```

Test Classification Report After Smote KNN Model

- By Looking the Smote and normal, Naive bayes model is much better compared to all other models. and as this is exit poll of election we can use method based on probabilities, result accuracy is much better.

1.6) Model Tuning (4 pts) , Bagging (1.5 pts) and Boosting (1.5 pts). Apply grid search on each model (include all models) and make models on best_params. Define a logic behind choosing particular values for different hyper-parameters for grid search. Compare and comment on performances of all. Comment on feature importance if applicable. Successful implementation of both algorithms along with inferences and comments on the model performances.

Solution:

Build Decision tree model:

We can fit our train-test spilt on Decision Tree Model and see results how it performs.

Fit Data in Decision Tree Classifier with Hyper Parameters which is used after model Tuning.

```
tree=DecisionTreeClassifier(criterion='gini',max_depth=3,random_state=1)
tree.fit(X_train,y_train)
```

```
DecisionTreeClassifier(max_depth=3, random_state=1)
```

```
print(tree.score(X_train,y_train))
print(tree.score(X_test,y_test))
```

```
0.8256794751640113
0.8056768558951966
```

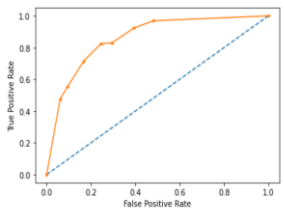
Criterion Used is Gini index.

Now We can check what are important features according to decision tree Classifier.

	Imp
age	0.000000
economic.cond.national	0.013385
economic.cond.household	0.000000
Blair	0.158513
Hague	0.492402
Europe	0.231168
political.knowledge	0.104532
gender	0.000000

As this model is not good for this problem , it is not showing accurate importance of variables , we can try different model to cross check variable importance.

```
[[297  31]
 [ 58  72]]
              precision    recall  f1-score   support

           0       0.84      0.91      0.87       328
           1       0.70      0.55      0.62       130

    accuracy                           0.81       458
   macro avg       0.77      0.73      0.74       458
weighted avg       0.80      0.81      0.80       458
```



ROC Curve for Test Data

Bagging Model:

Bagging, also known as bootstrap aggregation, is **the ensemble learning method that is commonly used to reduce variance within a noisy dataset**.

We had fit model by using grid search model tuning and used Hyper Parameters for Model as per Best_params

```
search_space={'n_estimators':[10,50,100,200],
              'max_samples':[0.5,1.0,1.5,2.0],
              'max_features':[0.5,1.0,1.5,2.0],
              'bootstrap':['True','False']}
```

```
from sklearn.model_selection import GridSearchCV
GS= GridSearchCV(estimator=bgclGS,
                 param_grid=search_space,
                 cv=5,
                 verbose=2,
                 n_jobs=-1)
```

We got hyper parameters as per best_params after applying Grid Search on Training Model.

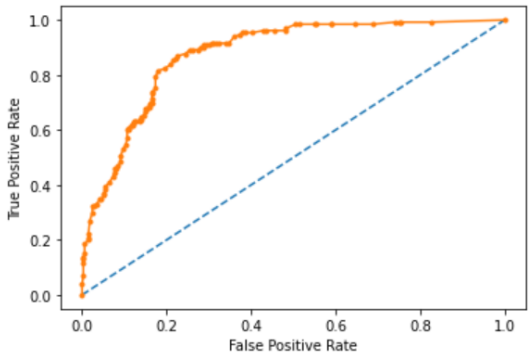
```
GS.best_params_
```

```
{'bootstrap': 'True',
 'max_features': 1.0,
 'max_samples': 0.5,
 'n_estimators': 100}
```

After fitting Data, we can see we are getting Train Score Approx 1 and test score approx. 0.80, It is not a good model because we are getting much good training model compare to test model.

```
[[283  45]
 [ 47  83]]
              precision    recall  f1-score   support

           0       0.86      0.86      0.86       328
           1       0.65      0.64      0.64       130

    accuracy                           0.80       458
   macro avg       0.75      0.75      0.75       458
weighted avg       0.80      0.80      0.80       458
```

Test Data Classification Report



ROC Curve for Test Model

AUC Score for Bagging Model:

Train Model: 0.99 , Test Model: 0.87

Boosting model:

For Boosting model, we will use Ada Boosting Classifier and Random Forest Classifier:

The basic concept behind Adaboost is to set the weights of classifiers and training the data sample in each iteration such that it ensures the accurate predictions of unusual observations.

Fitting Adaboosting model in train and test Data set after Find best hyper parameters from Grid Search method.

Model Tuning:

```
adclgv=AdaBoostClassifier(random_state=1)
```

```
parameters={'n_estimators':[10,50,100,150],
    'learning_rate':[0.001,0.01,0.1,1.0],
    'algorithm':['SAMME', 'SAMME.R']}
```

```
GSB=GridSearchCV(estimator=adclgv,
                 param_grid=parameters,
                 verbose=2,
                 cv=5,
                 n_jobs=-1)
```

Best params we got as following:

```
GSB.best_params_
```

```
{'algorithm': 'SAMME', 'learning_rate': 1.0, 'n_estimators': 100}
```

N_estimators : 100 , which means we can give hyper parameters of 100 iteration to calculate weight of classifiers when data is unusual observations.

Adaboosting Classifier helps us when the data is noisy or when we didn't have data in normally distributed form, It doesn't take in account outliers or null values , it will add weight to FP and FN predictions in every iteration and try to add weight and minimize the them.

Apply fit model on Test and train Model:

```
adcl=AdaBoostClassifier(n_estimators=100,random_state=1)
adcl.fit(X_train,y_train)
print(adcl.score(X_train,y_train))
print(adcl.score(X_test,y_test))
```

```
0.8472352389878163
0.8187772925764192
```

We can see the accuracy of train model: 0.84 and test model : 0.81

```
[[285  43]
 [ 40  90]]
              precision    recall  f1-score   support

           0       0.88      0.87      0.87       328
           1       0.68      0.69      0.68       130

    accuracy                           0.82       458
   macro avg       0.78      0.78      0.78       458
weighted avg       0.82      0.82      0.82       458
```

Classification Report for Test Model of Ada boosting Model.

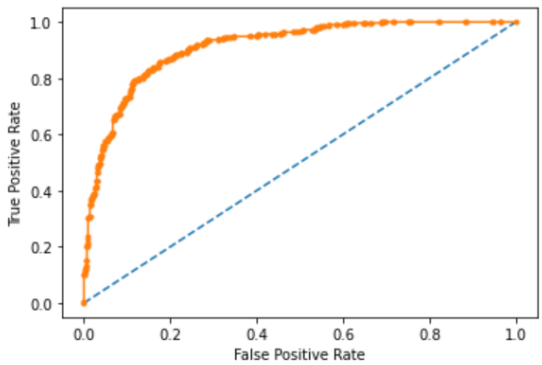
Model is performing Good on train and test Spilt.

Feature of importance as per Ada Boosting model:

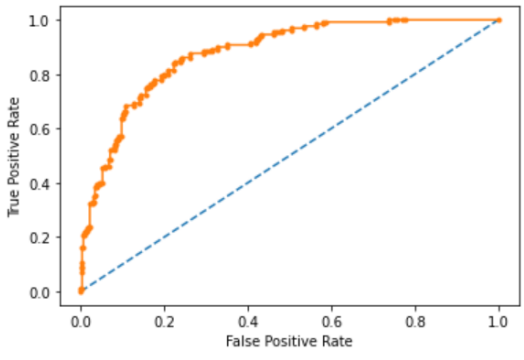
	Imp
age	0.69
economic.cond.national	0.04
economic.cond.household	0.03
Blair	0.06
Hague	0.08
Europe	0.07
political.knowledge	0.02
gender	0.01

Age , Hague & Europe is the variables of importance for predicting Target variable.

ROC CURVE FOR ADABOOSTING MODEL:



Area Under Curve for Train Model: 0.91



Area Under Curve for Test Model: 0.87

Train and test model for Adaboosting both are performing good, we can use this model as prediction model.

Random Forest Model:

It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

Apply fit Model on train and test Data spilt:

```
RF_model=RandomForestClassifier(n_estimators=50,random_state=1,max_features=5)
RF_model.fit(X_train,y_train)
print(RF_model.score(X_train,y_train))
print(RF_model.score(X_test,y_test))
y_predict8=RF_model.predict(X_test)
print(confusion_matrix(y_test,y_predict8))
print(classification_report(y_test,y_predict8))
```

```
0.9971883786316776
0.8165938864628821
```

For Hyper Parameter we had done model tuning

Classification report for test model:

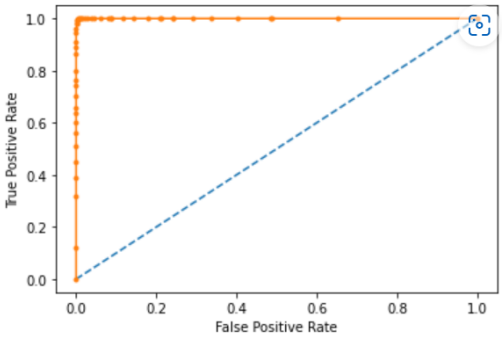
```
0.9971883786316776
0.8165938864628821
[[288  40]
 [ 44  86]]
              precision    recall  f1-score   support

           0       0.87      0.88      0.87       328
           1       0.68      0.66      0.67       130

    accuracy                           0.82       458
   macro avg       0.78      0.77      0.77       458
weighted avg       0.81      0.82      0.82       458
```

Accuracy for test:0.82
Accuracy for train:0.99
Precision & recall: 0.68 & 0.67
Classification report shows training model is 100 percent accurate compare to testing model 82 percent, this training model is to good compare to testing model we cannot use this model for prediction.

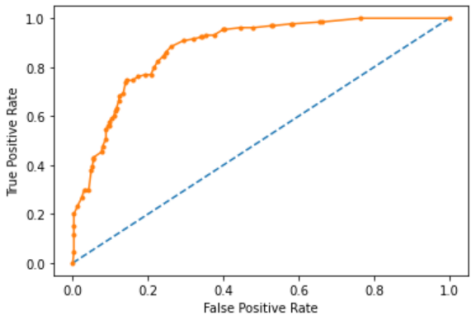
Important Features as per Random Forest Model:

	Imp
age	0.215123
economic.cond.national	0.073362
economic.cond.household	0.069183
Blair	0.131790
Hague	0.201555
Europe	0.184151
political.knowledge	0.095968
gender	0.028866

Age, Hague , Europe are sequential most important features to predict for target variable.



AUC Score for train Model: 0.99



AUC Score for train Model: 0.82

This model is not good to use as Training Model is giving more accuracy compared to testing Model.

1.7 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model, classification report (4 pts) Final Model - Compare and comment on all models on the basis of the performance metrics in a structured tabular manner. Describe on which model is best/optimized, After comparison which model suits the best for the problem in hand on the basis of different measures. Comment on the final model.

Soln:

Check Cross Validation for standard model:

It can reduce the size of data and ensures that the artificial intelligence model is robust enough.

```
from sklearn.model_selection import cross_val_score
```

```
from sklearn.utils import shuffle
_X,_Y=shuffle(X,y,random_state=1)
```

```
sc=cross_val_score(NNH,_X,_Y,cv=5)
```

```
sc.mean()
```

```
0.7895081967213116
```

```
sc.std()
```

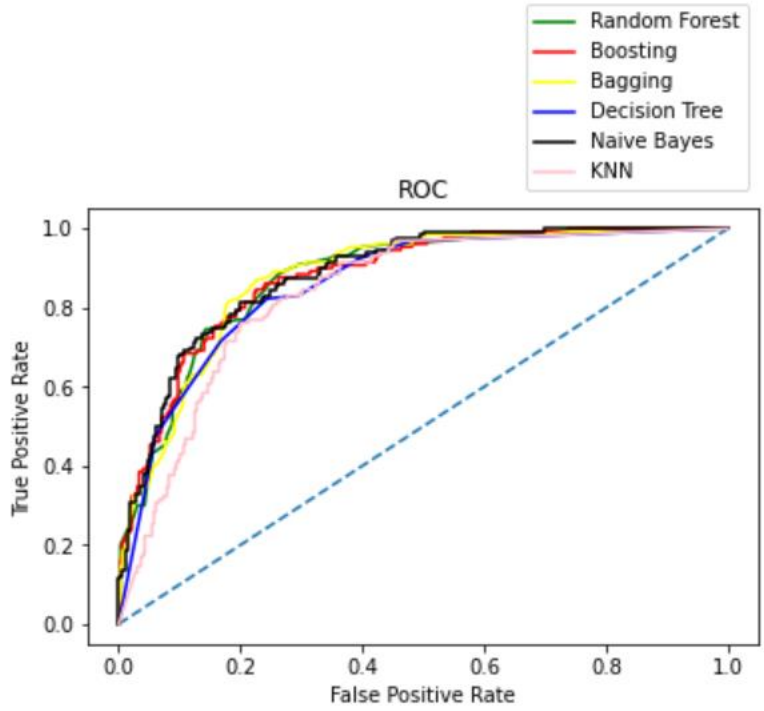
```
0.025430451711036445
```

so accuracy can be 78.9+-/0.2

Above Cross Validation score by keeping Kfold=5, we get Score mean: 0.78 with Deviation of 0.025.
Accuracy Score of: 78.9+- / 0.2.
Naïve Bayes and AdaBoosting Model are getting this train Accuracy Score, we can Say this two model can best fit for predicting target variable.

Comparing all models test Receiver Operating Curve:

```
plt.plot([0,1],[0,1],linestyle='--')
plt.plot(fpr_ran_test, tpr_ran_test,color='green', label='Random Forest')
plt.plot(fpr_ada_test,tpr_ada_test,color='red', label='Boosting')
plt.plot(fpr_bgcl_test,tpr_bgcl_test,color='yellow', label='Bagging')
plt.plot(fpr_tree_test,tpr_tree_test,color='blue', label='Decision Tree')
plt.plot(fpr_nav_test,tpr_nav_test,color='black', label='Naive Bayes')
plt.plot(fpr_knn_test,tpr_knn_test,color='pink', label='KNN')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC')
plt.legend(bbox_to_anchor=(0.,1.02,1.,.102),loc='lower right')
plt.show()
```



Boosting and Naïve Bayes Algorithm Models are having most inclination towards 1 compared to all other curves.

Generate Tabular form for accuracy, precision, recall & F1 Score for all Models:

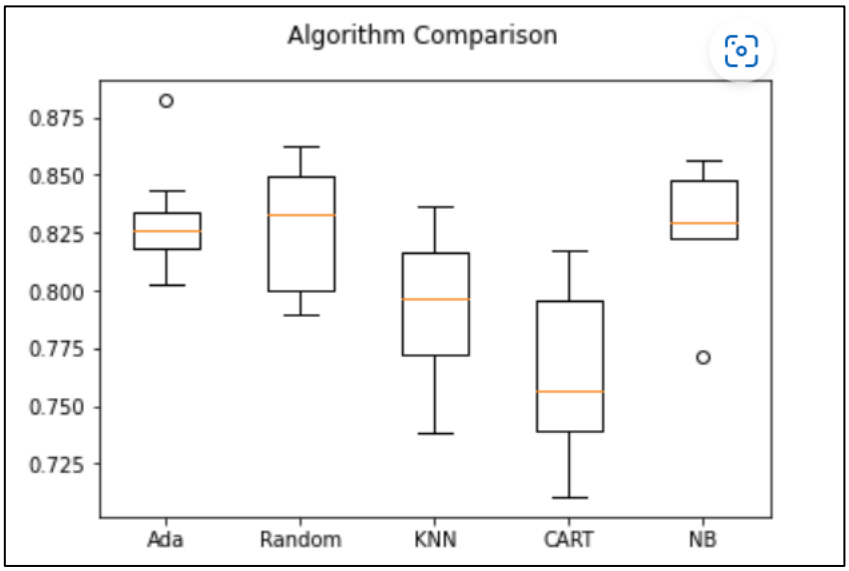
We will Compare all model on basis of precision and recall for test and train Data:

```
metric=np.array([[KNN_recall_test,KNN_precision_test,KNN_f1_test],
                 [NAV_recall_test,NAV_precision_test,NAV_f1_test],
                 [tree_recall_test,tree_precison_test,tree_f1_test],
                 [Bagging_recall_test,Bagging_precision_test,Bagging_f1_test],
                 [AdaBoosting_recall_test,AdaBoosting_precision_test,AdaBoosting_f1_test],
                 [Random_recall_test,Random_precision_test,Random_f1_test],
                 [LDA_recall_test,LDA_precision_test,LDA_f1_test],
                 [LOG_recall_test,LOG_precision_test,LOG_f1_test]
                ]).T
```

	Model	Accuracy	Recall	Precision	F1_Score
0	KNN Model	0.781659	0.607692	0.617188	0.612403
1	Naive_bayes	0.825328	0.723077	0.681159	0.701493
2	Decision_tree	0.805677	0.553846	0.699029	0.618026
3	Bagging	0.799127	0.638462	0.648438	0.643411
4	AdaBoosting	0.818777	0.692308	0.676692	0.684411
5	Random_Forest	0.816594	0.661538	0.682540	0.671875
6	LDA	0.818777	0.661538	0.688000	0.674510
7	Logistics_Regression	0.823144	0.702479	0.653846	0.677291

Categorize problem by output:

The output of the model is a class, it's a classification problem.



By Algorithm Comparison we can see Boosting Model and Naïve Bayes Model are good Model for Prediction of Target Variable.

Cross Validation Accuracy Score with Std deviation for Algorithms:

```
Ada Boost: 0.828836 (0.021520)
Random Forest: 0.825585 (0.024129)
KNN: 0.794092 (0.030513)
CART: 0.769788 (0.034508)
Naive Bayes: 0.830181 (0.023541)
```

We can see , getting highest accuracy in Boosting and Naïve Bayes , Naïve Bayes is algorithm of Conditional Probability this model is accurate for this type of problem , but as data will get large , we need to switch on Boosting Models, as naïve bayes model is good for short Dataset .

1.8) Based on your analysis and working on the business problem, detail out appropriate insights and recommendations to help the management solve the business objective. There should be at least 3-4 Recommendations and insights in total. Recommendations should be easily understandable and business specific, students should not give any technical suggestions.

- **Age is most important factor in voting for labour or conservative party.**
- **Young Age Group of 30 to 60 is voting more for Labour Party**
- **Hague is Conservative leader currently who has more impactful assessment because of his support to European Union, People are leaning towards party who is favouring Eurosceptic nature.**
- **People's political knowledge for parties stand on European integration plays important role in voting.**
- **Hague Conservative leader tends to have more impact than Blair Labour Party Leader in Exit polls, but his needs to present economic reforms policy for household and national both as voters still don't have much emphasis on Economic matters.**
- **Hague Conservative Party Leader should target young age population and share manifesto stating his strong stand on European integration and economic reforms.**
- **Highest population of voters are young age people, party manifesto must made on economic reforms like employments plan , cheaper Housing.**
- **European integration is the process of industrial, economic, political, legal, social, and cultural integration of states wholly or partially in Europe or nearby. Which is important factor in voting for voters, Party must promote his stances on economic, social and cultural collaboration with other European countries.**
- **Ease of Migration policy must be presented with better opportunities for people.**

Problem 2:

In this 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.

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

Import important libraries.

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

Create Data frame from Series or raw Data Inaugural.

```
import pandas as pd

from nltk.corpus import inaugural as presidents

president = []
for fileid in presidents.fileids():
    tag, filename = fileid.split('-')
    president.append((filename, tag, presidents.raw(fileid)))

df = pd.DataFrame(president, columns=['President', 'year', 'text'])
```

Download Stopwords and Tokenizer Word from nltk.corpus library.

```
1 import nltk
2 from nltk.corpus import stopwords
3 from nltk.tokenize import word_tokenize,sent_tokenize
4 import matplotlib.pyplot as plt
5 import string
6 import re
```

Count of Words without removing Stopwords from 3 President Speech:

	President	text	Word_count
38	Roosevelt.txt	On each national day of inauguration since 178...	1323
43	Kennedy.txt	Vice President Johnson, Mr. Speaker, Mr. Chief...	1364
46	Nixon.txt	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	1769

President Nixon had longest speech among 3 presidents.

Number of Sentences for 3 president speeches are 68 ,52 ,69 respectively.

2.2) Remove all the stopwords from the three speeches. Show the word count before and after the removal of stopwords. Show a sample sentence after the removal of stopwords.

We can remove stopwords by using stopwords library from corpus and set lang as English,
Other punctuation and special Characters we can add in list of Stopwords.

Along with removing Stopwords, we need to convert whole text into Lower Case.

By Apply Lambda Function we can get above desired Output.

```
# Add stopwords
stop_words=stopwords.words('english')
add_to_stop_words=[',','.','--','?','(',')','-',';',"'",";",]
stop_words.extend(add_to_stop_words)
stop_words=set(stop_words)
```

```
result['Stopwords']=result['text'].apply(lambda x: len([x for x in x.split() if x in stop_words]))
result[['text','Stopwords']].head()
```

Converting Whole Text in Lower Case by Lambda Function:

```
result['text'] = result['text'].apply(lambda x: " ".join(x.lower() for x in x.split()))
result['text'].head()
```

We can Compare Count of Words Before and After Removal of Stop Words:

President	year	text	Word_count	Stopwords	Word_count_Aftr_StopWords
oosevelt.txt	1941	national day inauguration since 1789 people re...	1323	654	627
Kennedy.txt	1961	vice president johnson mr speaker mr chief jus...	1364	642	693
Nixon.txt	1973	mr vice president mr speaker mr chief justice ...	1769	916	833

Sample Sentence After Removing Stopwords:

```
print(result.iloc[0:1]['text'])
```

```
  national day inauguration since 1789 people re...
```

2.3) Which word occurs the greatest number of times in his inaugural address for each president? Mention the top three words. (after removing the stopwords)

We can get top three words for each President Speeches by Converting into Series and applying Join Function.

President Roosevelt:

```
freqRoosevelt=pd.Series(" ".join(result.iloc[0:1]['text']).split()).value_counts()[:3]
freqRoosevelt
```

```
nation       11
know         10
democracy     9
```

President Kennedy:

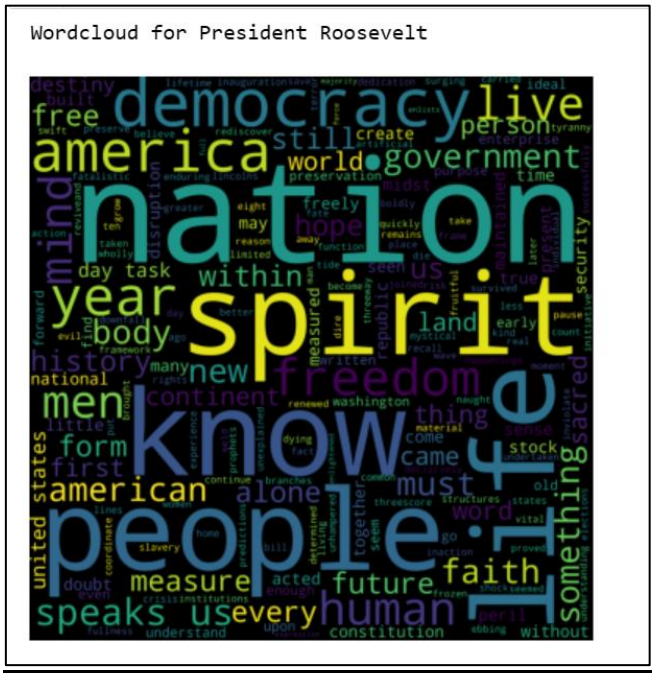
```
let       16
us        12
sides      8
```

President Nixon:

Top three words in Speech.

```
us        26
let       22
peace     19
```

2.4) Plot the word cloud of each of the three speeches. (After removing the stopwords)
Word Cloud after Removing Stopwords for Each President.
Import Word Cloud Library.



Word Cloud for President Kennedy

Word Cloud for President Nixon

We can Drop words and check the Word Cloud after unimportant Words.