Machine Learning Project

Name: Swetha Kunapuli

Batch & Course: PGP-DSBA

Online June Batch

Date: 07/1/2022

Table of Contents

List of Tables:

List of Figures:

List of Screenshots:

PROBLEM 1

Executive Summary:

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 must build a model, to predict which party a voter will vote based on the given information, to create an exit poll that will help in predicting overall win and seats covered by a particular party.

Election_Data.xlsx

Data Introduction:

The purpose of this whole exercise is to explore the dataset and is recommended for learning and practicing our skills using machine learning model.

The dataset contains 1525 rows and 9 columns.

Data Description:

Description of variables is as follows:

Variable Name	Description
Vote(categorical)	Party Choice: Conservative or Labour
Age(numerical)	In Years
Economic.cond.national(numerical)	Assessment of current national economic condition, on scale 1 to 5.
Economic.cond.household(numerical)	Assessment of current household economic conditions, on scale 1 to 5.
Blair(numerical)	Assessment of the Labour leader, on scale 1 to 5.
Hague(numerical)	Assessment of the Conservative leader, on scale 1 to 5.
Europe(numerical)	11-point scale that measures respondents' attitude towards European integration.
Political.knowledge(numerical)	Knowledge of parties' position in Europe
Gender(categorical)	Female/Male

Table 1: Variables Information

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

Table 2: Description of Variables

Sample of the dataset:

	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

Table 3: Head of the dataset

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
1520	Conservative	67	5	3	2	4	11	3	male
1521	Conservative	73	2	2	4	4	8	2	male
1522	Labour	37	3	3	5	4	2	2	male
1523	Conservative	61	3	3	1	4	11	2	male
1524	Conservative	74	2	3	2	4	11	0	female

Table 4: Tail of the dataset

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

Exploratory Data Analysis:

Check for types of variables in the data frame:

```
<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
```

Table 5: Dataset Information

Observation:

Dataset has 9 variables and 1525 records. We have int and object data types in the given dataset.

Check for missing values in the dataset:

```
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
```

Table 6: Missing values information

Observation:

There are no missing values in the given dataset.

Check for duplicate values in the dataset:

```
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

Table 7: Duplicate rows in the dataset

Observation:

There are 8 duplicate rows in the dataset and removed duplicate values.

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

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

Table 8: Duplicate values information

Check for unique values for categorical variables

```
VOTE   2
Conservative     460
Labour          1057
Name: vote, dtype: int64
GENDER   2
male      709
female    808
Name: gender, dtype: int64
```

Table 9: Unique categorical variables

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

Univariate Analysis:

Vote

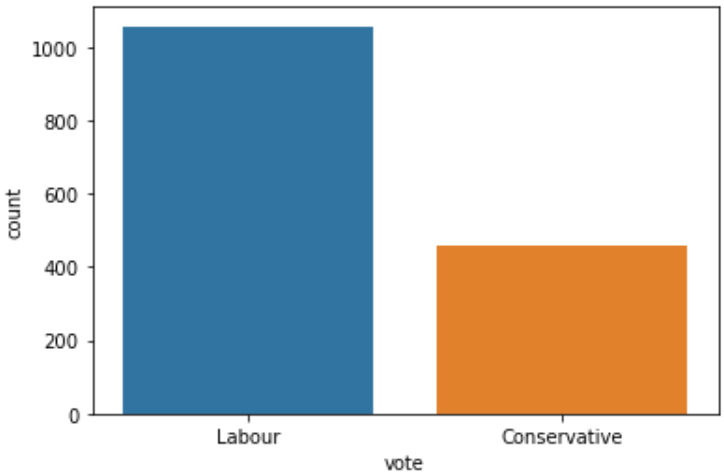

Figure 1: Box plot – Vote variable

Vote is the target variable.

The above plot describes the number of candidates who vote for labour is more than the conservative party.

Age

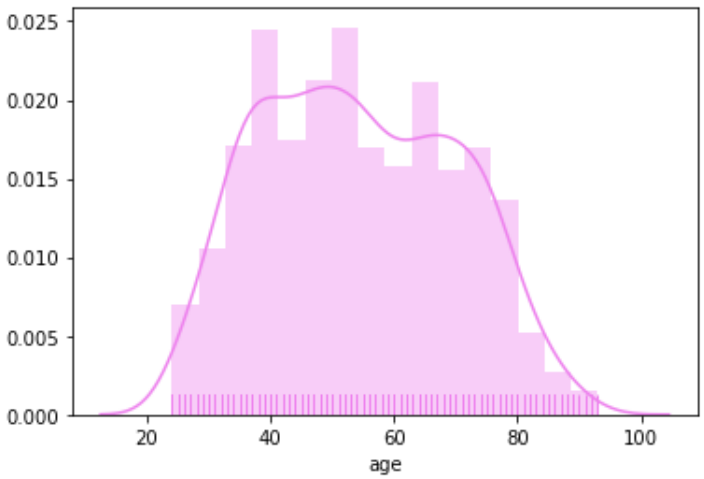

Figure 2: Distplot – Age variable

Age: The above plot describes the age group of the candidates who vote.

Economic.cond.national

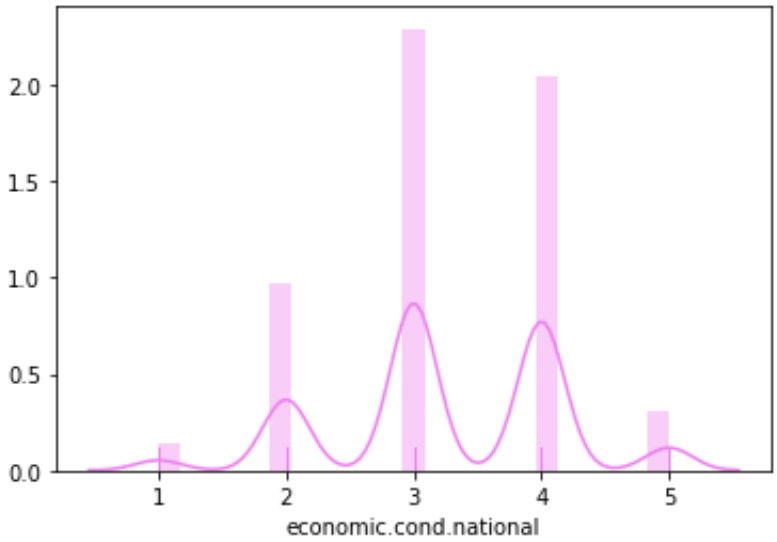

Figure 3: Distplot – Economic.cond.national variable

Economic.cond.national: The above plot describes the economic condition of the nation which falls between scale1-5.

Economic.cond.household

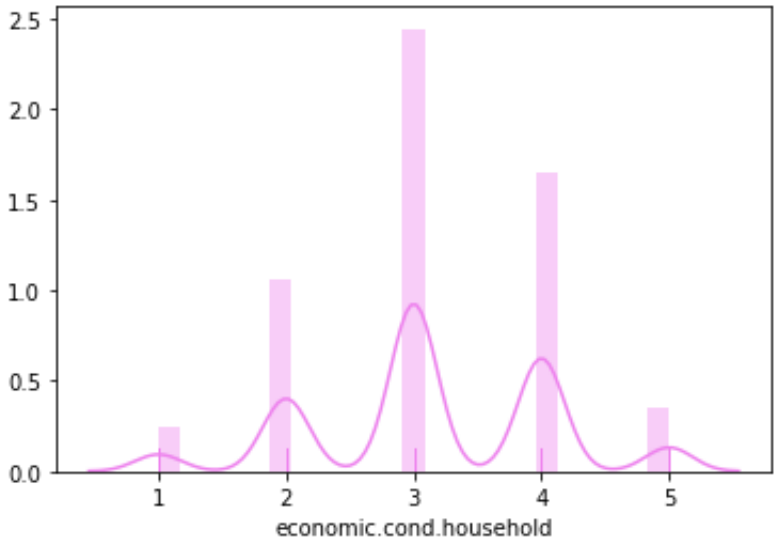

Figure 4: Distplot – Economic.cond.household variable

Economic.cond.household: The above plot describes the economic condition of household, which also falls between scale 1-5.

Blair

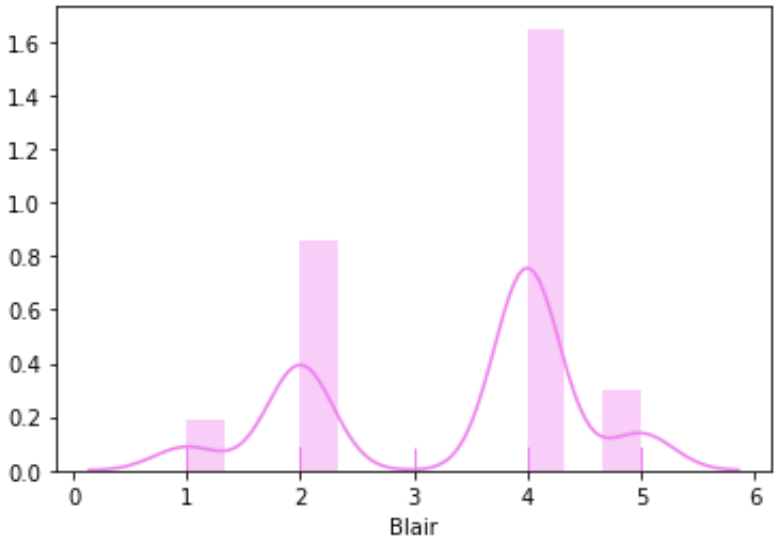

Figure 5: Distplot – Blair variable

Blair: It is the Assessment of the Labour leader on a scale 1-5

Hague

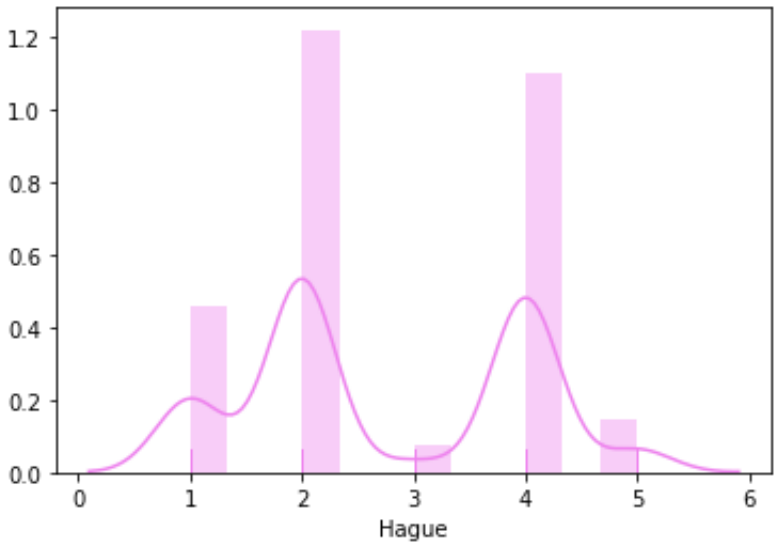

Figure 6: Distplot – Hague variable

Hague: It belong to conservative party on scale of 1-5

Europe

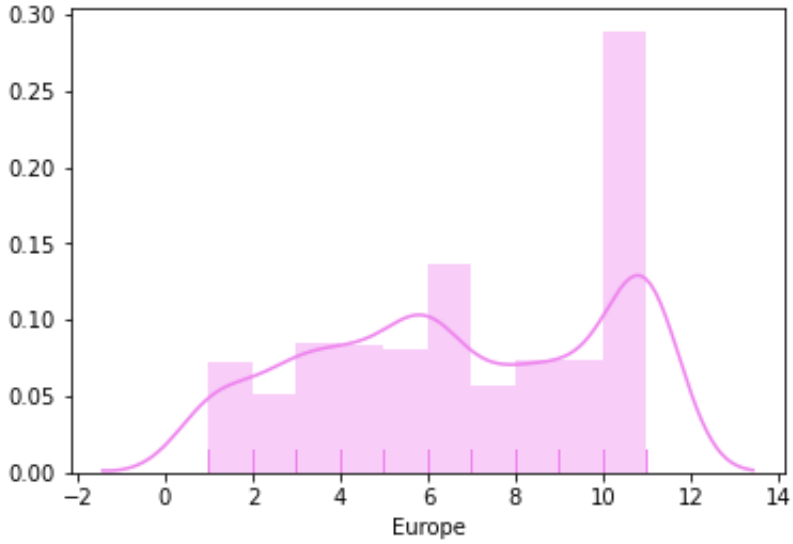

Figure 7: Distplot – Europe variable

Political.knowledge

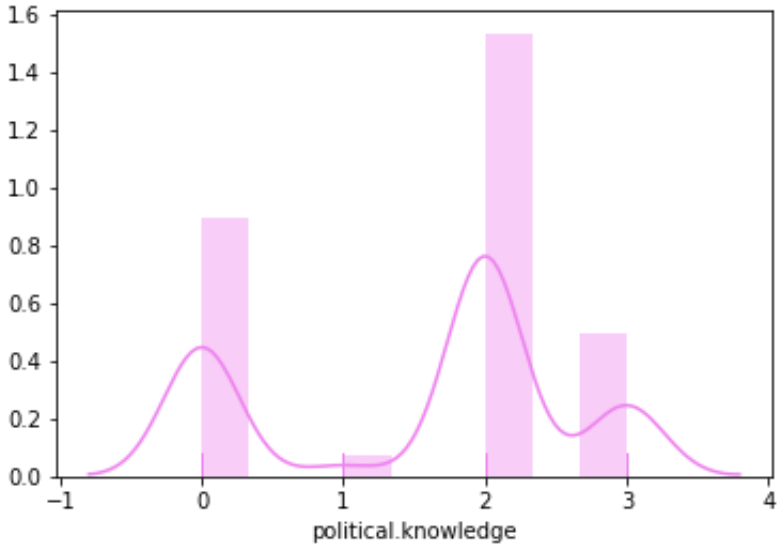

Figure 8: Distplot – Political.knowledge variable

Political.knowledge: The above plot describes the knowledge of party's position

Gender

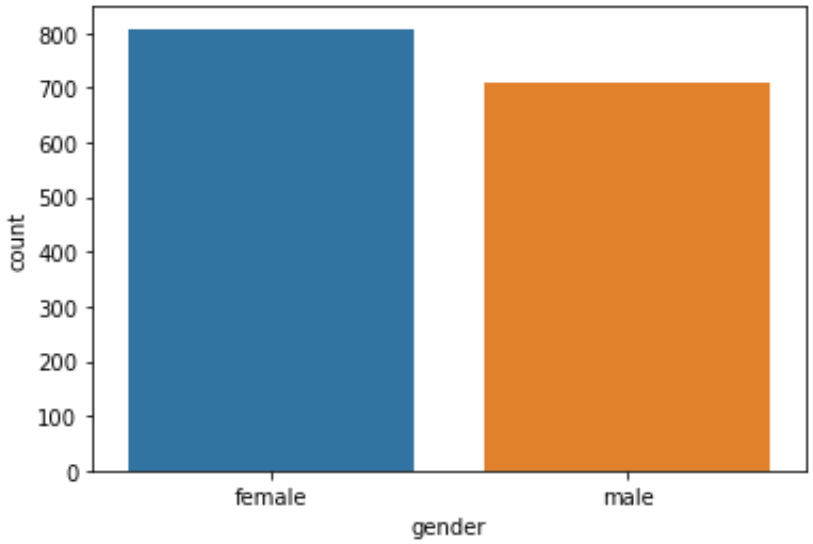

Figure 9: Box plot – Gender variable

Gender: The above plot describes the number of female and male candidates who has voted for the parties

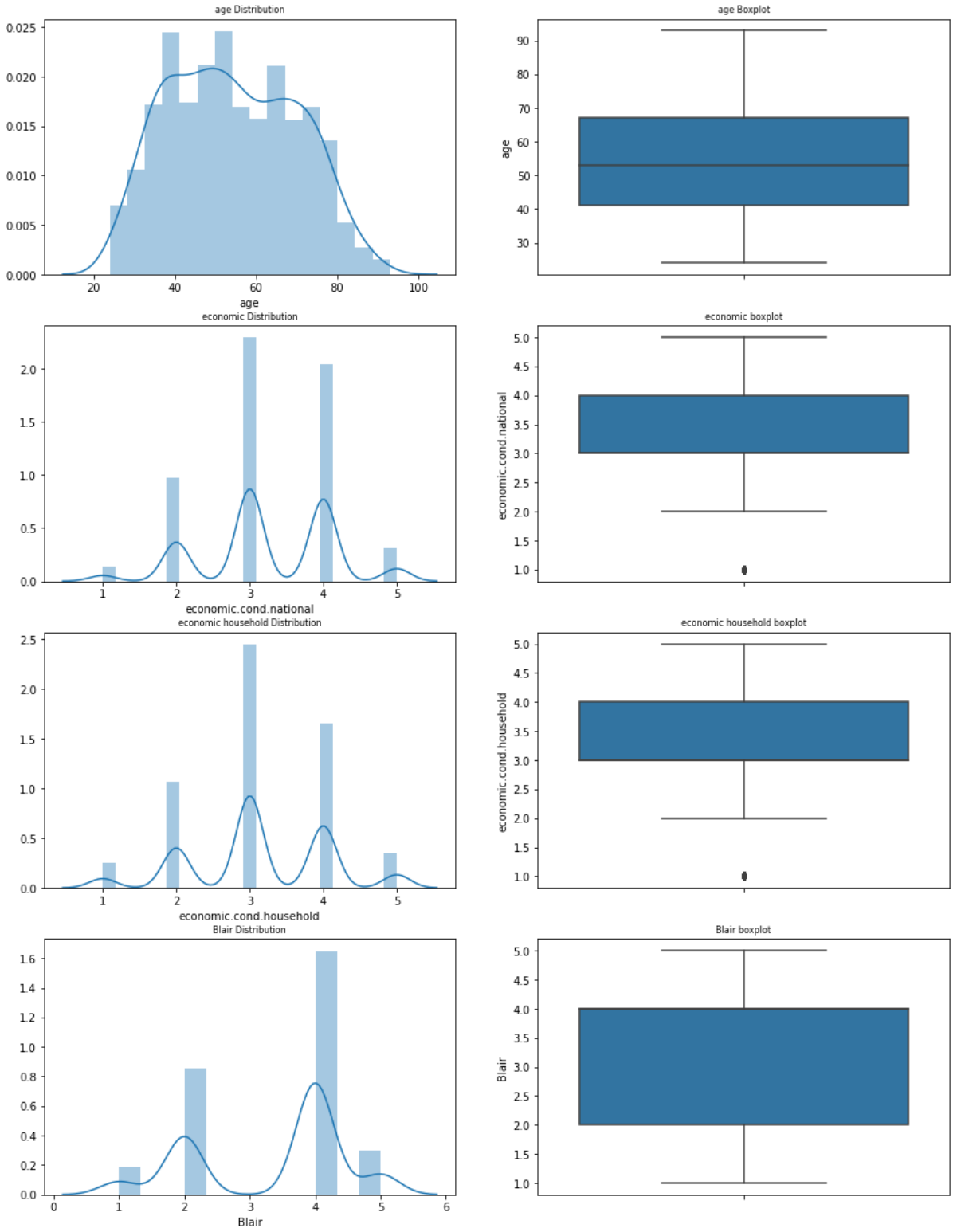

Figure 10: Dist & Box plots – All variables

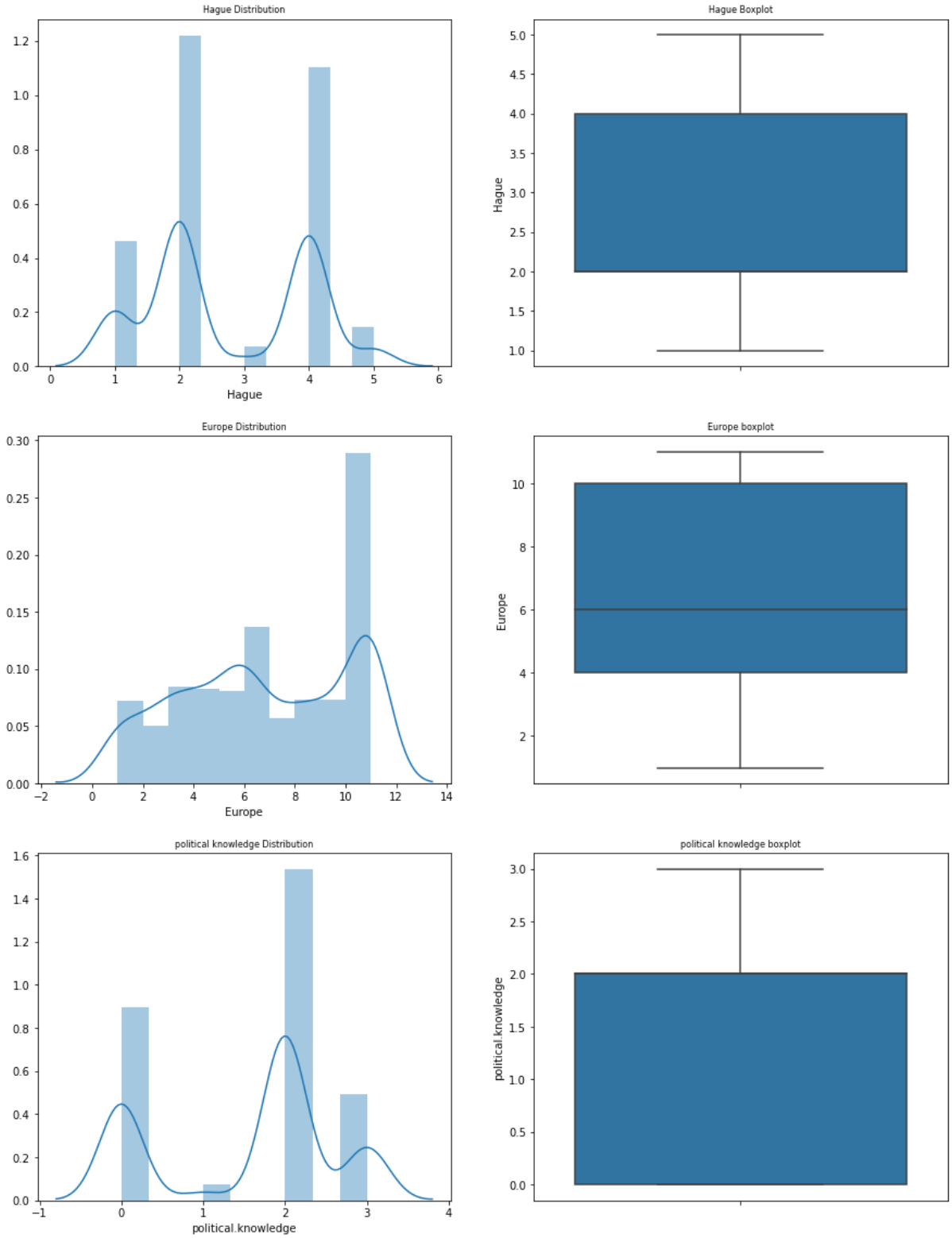

Figure 11: Dist & Box plots – All variables

We can see that all the numerical variables are normally distributed (not perfectly normal though and are multi modal in some instances). There are outliers present in “economic.cond.national” and ”economic.cond.household”.

Bivariate Analysis:

Vote vs Age

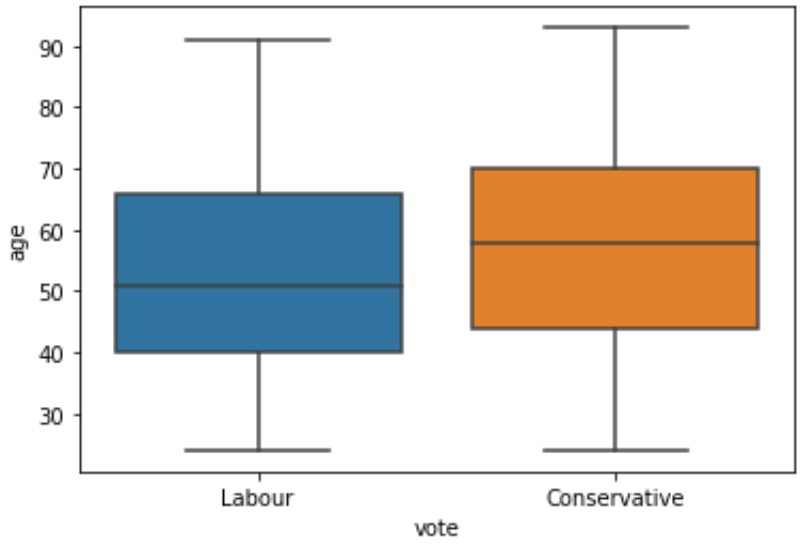

Figure 12: Box plot – Vote vs Age

From the above plot we can see people with age group between 40-45 has not voted to Conservative part and people with age group between 65-70 has not voted to Labour party.

Vote vs Economic.cond.national

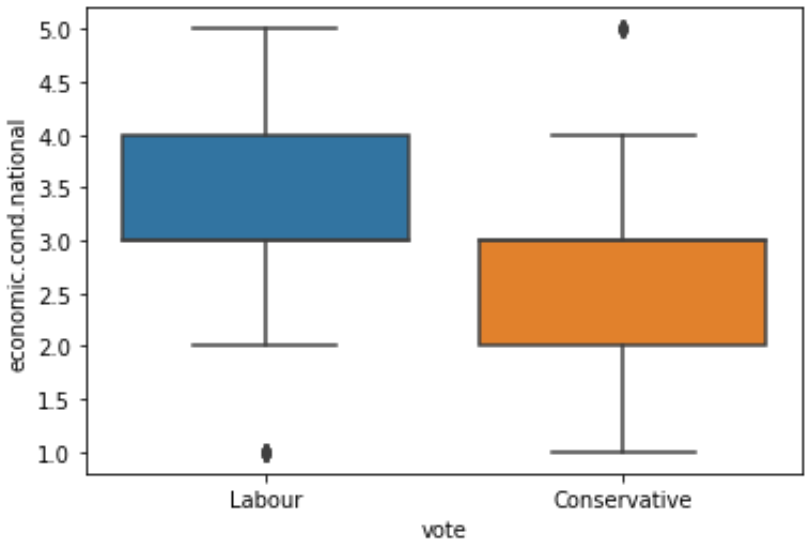

Figure 13: Box plot – Vote vs Economic.cond.national

From the above plot we can see that the economic condition of nation is good under Labour party ruling compared to Conservative party.

Vote vs Economic.cond.household

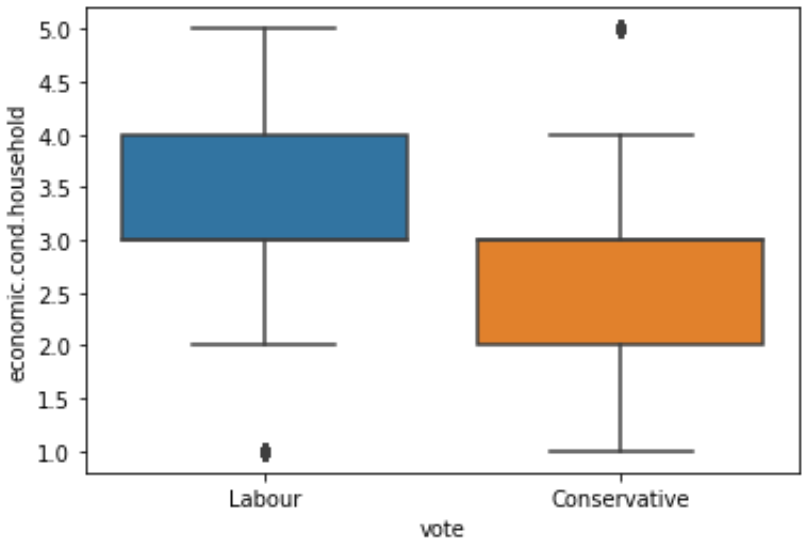

Figure 14: Box plot – Vote vs Economic.cond.household

From the above plot we can see that the economic condition of household is good under Labour party ruling compared to Conservative party.

Vote vs Blair

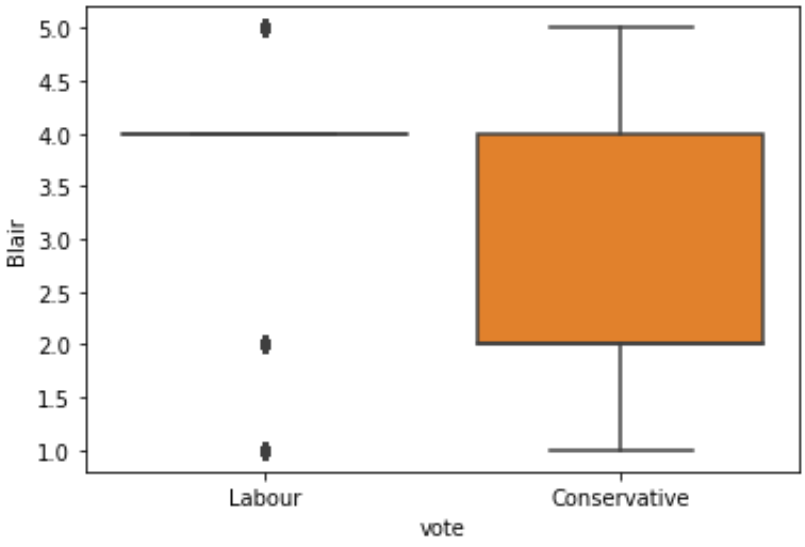

Figure 15: Box plot – Vote vs Blair

From the above plot we can see Labour party is assessed with scaling 4.0.

Vote vs Hague

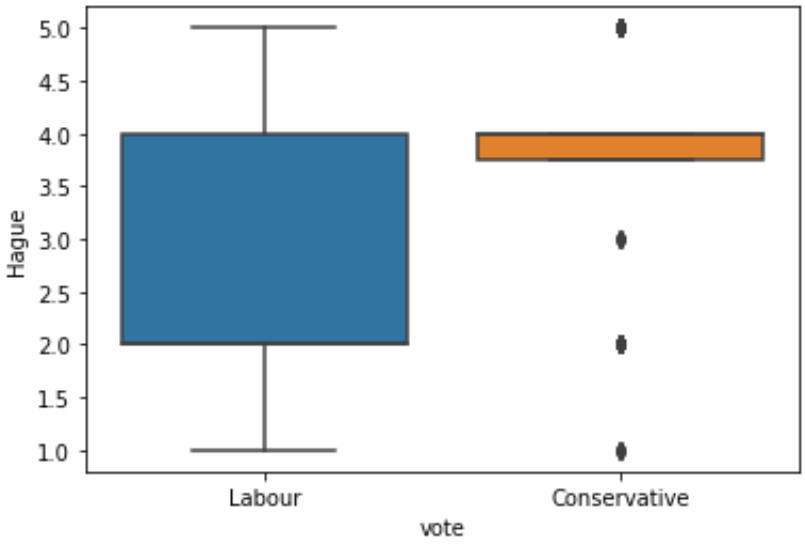

Figure 16: Box plot – Vote vs Hague

From the above plot we can see Conservative party is assessed with scaling between 3.5 and 4.0.

Vote vs Europe

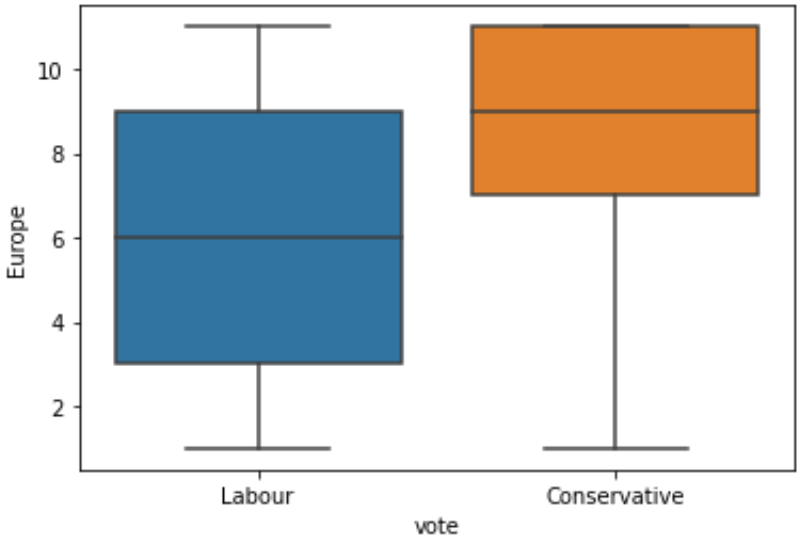

Figure 17: Box plot – Vote vs Europe

From the above plot we can see Labour party is strong party when compared to Conservative party in Europe region.

Vote vs Political.knowledge

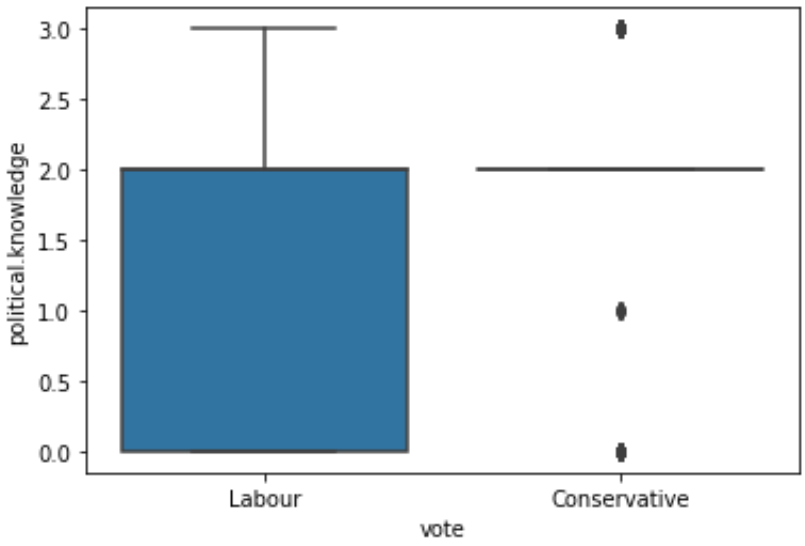

Figure 18: Box plot – Vote vs Political.knowledge

From the above plot we can see Labour part has more political knowledge when compared to Conservative party.

Outliers:

There are outliers present in the categorical variable vote and in numeric fields – economic.cond.national and economic.cond.household.

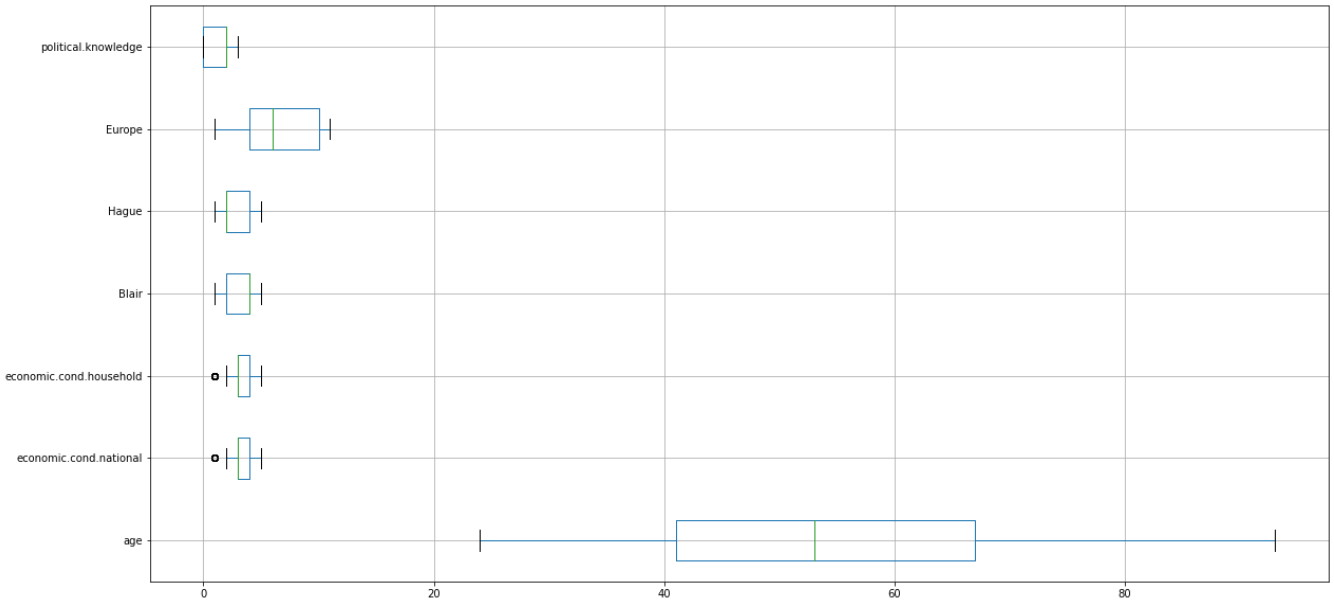

Figure 19: Box plot – All variables

After treating outliers, below are the plots and data description

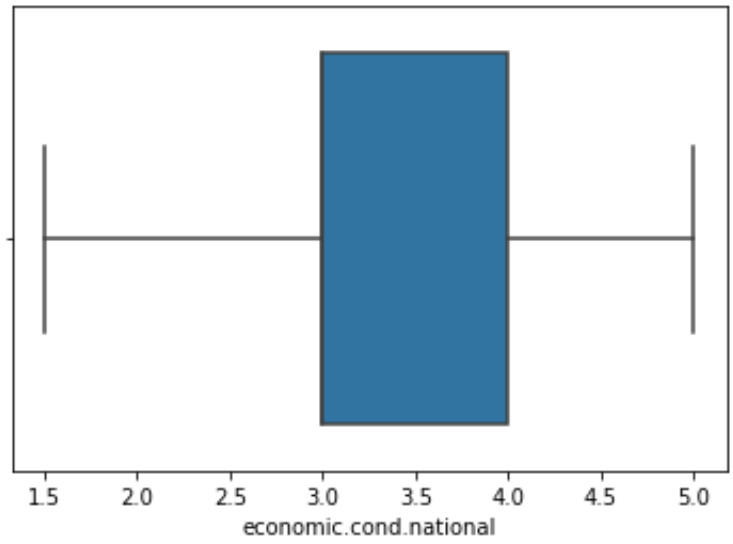

Figure 20: Box plot – Economic.cond.national

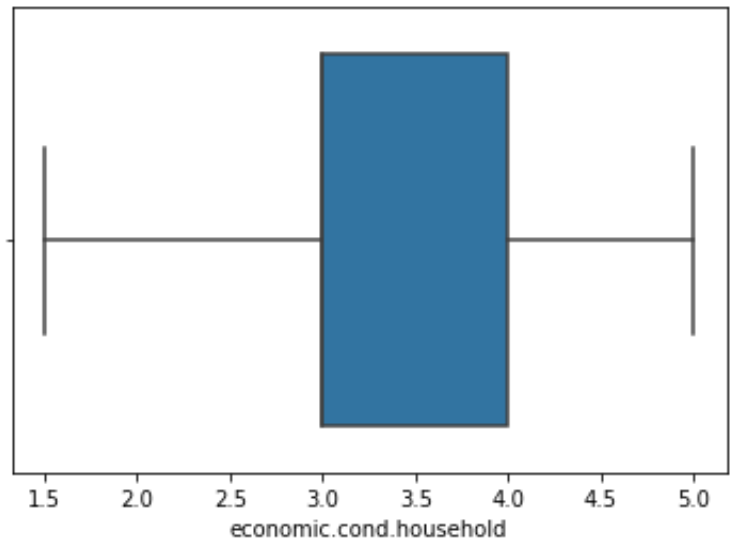

Figure 21: Box plot – Economic.cond.household

	count	mean	std	min	25%	50%	75%	max
age	1517.0	54.241266	15.701741	24.0	41.0	53.0	67.0	93.0
economic.cond.national	1517.0	3.257416	0.853647	1.5	3.0	3.0	4.0	5.0
economic.cond.household	1517.0	3.159196	0.886279	1.5	3.0	3.0	4.0	5.0
Blair	1517.0	3.335531	1.174772	1.0	2.0	4.0	4.0	5.0
Hague	1517.0	2.749506	1.232479	1.0	2.0	2.0	4.0	5.0
Europe	1517.0	6.740277	3.299043	1.0	4.0	6.0	10.0	11.0
political.knowledge	1517.0	1.540541	1.084417	0.0	0.0	2.0	2.0	3.0

Table 10: Data description

Data Distribution

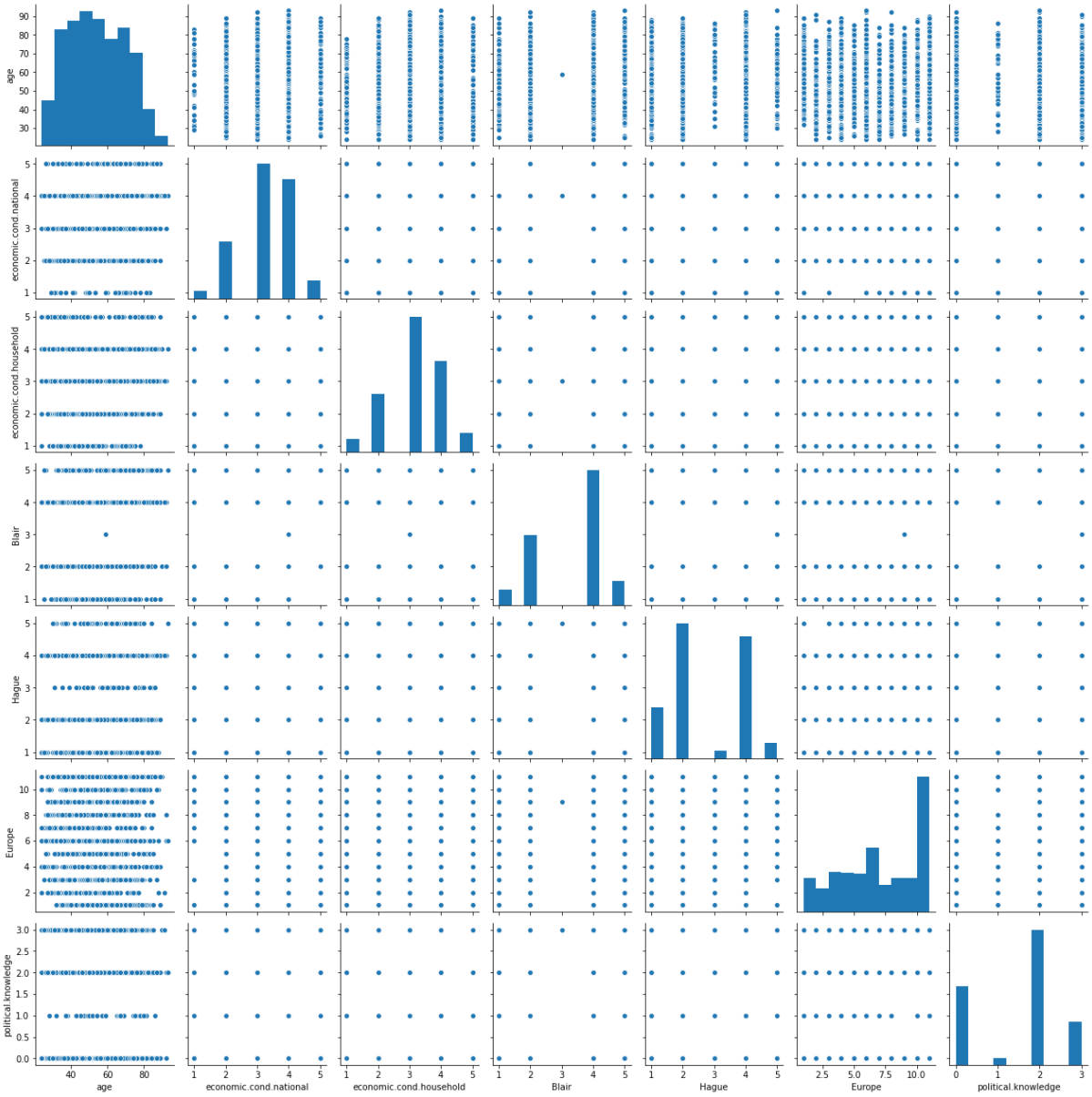

Figure 22: Pair plot – All variables

Pair plot tells us about interaction of each variable with every other variable present.

As per above plot, there is no strong relationship present between the variable.

There is a mixture of positive and negative relationships though which is expected.

Correlation Matrix

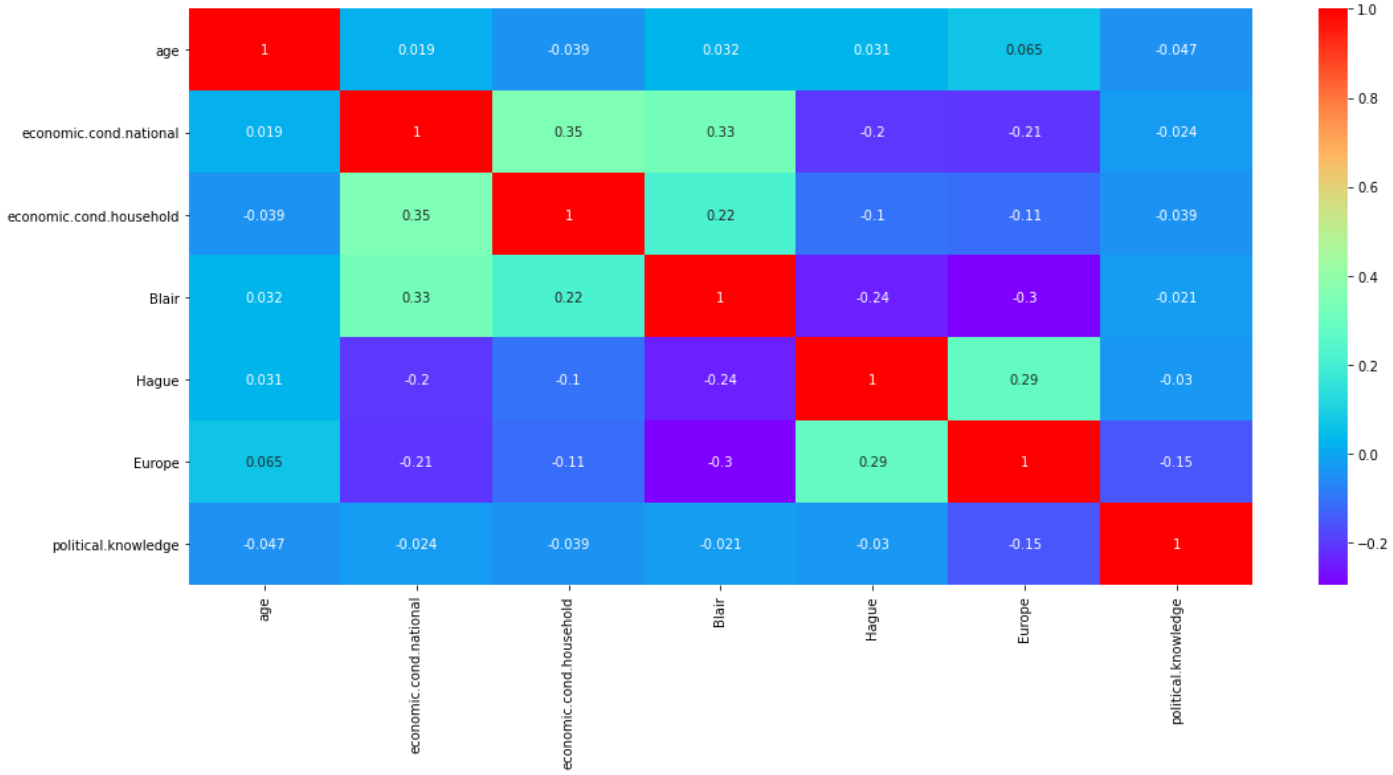

Figure 23: Heat map – All variables

Multicollinearity is an important issue which can harm the model. Heatmap is a good way of identifying this issue. It gives us a basic idea of relationship the variables have with each other.

Observations:

Highest positive correlation is between “economic_cond_national” and “economic_cond_household” which is 35% and it is not huge.

Highest negative correlation is between “Blair” and “Europe” which is 30% and it is also not huge.

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).

Encoding the data:

```
<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
```

Table 11: Data information

Vote and Gender variables are integer data types, and we need to convert them to object data type to perform operations.

For encoding we need to convert the categorical string value to categorical numeric values.

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	vote_Labour	gender_male
0	43	3.0	3.0	4	1	2	2	1	0
1	36	4.0	4.0	4	4	5	2	1	1
2	35	4.0	4.0	5	2	3	2	1	1
3	24	4.0	2.0	2	1	4	0	1	0
4	41	2.0	2.0	1	1	6	2	1	1

Table 12: Sample of the data encoded

Vote has 2 variables: Conservative and Labour, converted them to Conservative as 0 and Labour as 1 using get dummies function as shown below.

```
1    1057
0     460
Name: vote_Labour, dtype: int64
```

Screenshot 1

Similarly Gender also has 2 variables: Male and Female, converted them to Female as 0 and Male as 1 using get dummies function as shown below.

```
0    808
1    709
Name: gender_male, dtype: int64
```

Screenshot 2

After performing encoding renamed column “vote_Labour” to “IsLabour_or_not”, in this variable 1 stands for Labour party, 0 stands for Conservative party.

Similarly renamed column “gender_male” to “IsMale_or_not”, in this variable 1 stands for Male and 0 stands for Female.

Sample of the dataset after performing encoding

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	IsLabour_or_not	IsMale_or_not
37	52	4.0	3.0	4	4	3	2	1	1
1334	38	3.0	1.5	2	4	7	2	0	1
351	83	2.0	4.0	2	4	10	2	0	0
889	35	3.0	4.0	2	4	11	0	0	1
745	70	2.0	2.0	4	2	10	2	0	0
1285	44	4.0	4.0	4	2	5	2	1	1
1066	66	3.0	3.0	2	4	6	2	1	1
103	70	2.0	3.0	2	4	8	2	0	0
479	71	5.0	3.0	5	2	2	2	1	0
1429	53	3.0	4.0	4	2	10	0	1	1

Table 13: Sample of the data after encoding

Scaling:

It totally depends on the model we are building whether scaling is required or not.

Usually, the distance-based methods(E.g.: KNN) would require scaling as it is sensitive to extreme difference and can cause a bias. But the tree-based methods(E.g.: Decision Tree) would not require scaling in general as it unnecessary since they use split method.

Here, I am performing scaling only for KNN model. Rest of the models scaling is not required.

Data Split: Splitting the data into test and train

Before splitting the data, we need to find target variable. Here “vote” is the target variable in the original given dataset and after performing encoding its renamed to “IsLabour_or_not”.

Hence my target variable is “IsLabour_or_not” and splitting data into 70:30 ratio.

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

Logistic Regression Model

```
model = LogisticRegression(solver='newton-cg',max_iter=100,penalty='none',verbose=True,n_jobs=2)
model.fit(X_train, y_train)
```

```
[Parallel(n_jobs=2)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=2)]: Done   1 out of   1 | elapsed:    1.3s finished
```

```
LogisticRegression(n_jobs=2, penalty='none', solver='newton-cg', verbose=True)
```

Screenshot 3

Train Data Accuracy Score

```
0.8341187558906692
```

Screenshot 4

Test Data Accuracy Score

```
0.8289473684210527
```

Screenshot 5

Grid Search CV on Logistic Regression

```
grid_search.fit(X_train, y_train)
```

```
GridSearchCV(cv=2, estimator=LogisticRegression(max_iter=1000, n_jobs=2),
             n_jobs=-1,
             param_grid={'penalty': ['l2', 'none'], 'solver': ['sag', 'lbfgs'],
                         'tol': [0.01, 0.001]},
             scoring='f1')
```

```
print(grid_search.best_params_,'\n')
print(grid_search.best_estimator_)
```

```
{'penalty': 'none', 'solver': 'lbfgs', 'tol': 0.01}

LogisticRegression(max_iter=1000, n_jobs=2, penalty='none', tol=0.01)
```

Screenshot 6

	precision	recall	f1-score	support
0	0.76	0.73	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

Table 14: Classification report – Logistic Regression test data

As we can see from the above screenshot 5 & table14 – the accuracy of test data score is same for logistic regression model with applying grid search cv method and without applying grid search cv method.

Linear Discriminant Analysis(LDA Model)

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
LDA_model= LinearDiscriminantAnalysis()
LDA_model.fit(X_train, y_train)
```

```
LinearDiscriminantAnalysis()
```

Screenshot 7

Train Data Accuracy Score

```
0.8341187558906692
```

Screenshot 8

Test Data Accuracy Score

```
0.831140350877193
```

Screenshot 9

Grid Search CV on LDA

```
X, y = make_classification(n_samples=1000, n_features=10, n_informative=10, n_redundant=0, random_stat
# define model
model = LinearDiscriminantAnalysis()
# define model evaluation method
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
# define grid
grid = dict()
grid['solver'] = ['svd', 'lsqr', 'eigen']
# define search
search = GridSearchCV(model, grid, scoring='accuracy', cv=cv, n_jobs=-1)
# perform the search
results = search.fit(X, y)
# summarize
print('Accuracy: %.3f' % results.best_score_)
print('Config: %s' % results.best_params_)
```

```
Accuracy: 0.893
Config: {'solver': 'svd'}
```

Screenshot 10

Next, we can explore whether using shrinkage with the model improves performance.

Shrinkage adds a penalty to the model that acts as a type of regularized, reducing the complexity of the model.

```
X, y = make_classification(n_samples=1000, n_features=10, n_informative=10, n_redundant=0, random_state
# define model
model = LinearDiscriminantAnalysis(solver='lsqr')
# define model evaluation method
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
# define grid
grid = dict()
grid['shrinkage'] = arange(0, 1, 0.01)
# define search
search = GridSearchCV(model, grid, scoring='accuracy', cv=cv, n_jobs=-1)
# perform the search
results = search.fit(X, y)
# summarize
print('Accuracy: %.3f' % results.best_score_)
print('Config: %s' % results.best_params_)
```

```
Accuracy: 0.894
Config: {'shrinkage': 0.02}
```

Screenshot 11

As we can see in the above screenshots 9, 10 & 11 - the accuracy score of the test data is same for LDA model using grid search cv and without using grid search cv method.

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

Naive Bayes Model

```
NB_model = GaussianNB()
NB_model.fit(X_train, y_train)
```

```
GaussianNB()
```

Screenshot 12

Train Data Accuracy Score

```
0.8341187558906692
```

Screenshot 13

Test Data Accuracy Score

```
0.8223684210526315
```

Screenshot 14

Grid Search CV on Naive Bayes Model

```
param_grid_nb = {
    'var_smoothing': np.logspace(0,-9, num=100)
}
nbModel_grid = GridSearchCV(estimator=GaussianNB(), param_grid=param_grid_nb, verbose=1, cv=10, n_jobs=
nbModel_grid.fit(X_train, y_train)
print(nbModel_grid.best_estimator_)
```

```
Fitting 10 folds for each of 100 candidates, totalling 1000 fits
GaussianNB(var_smoothing=0.0012328467394420659)
```

Screenshot 15

Accuracy score for test data after using grid search cv method

```
y_pred = nbModel_grid.predict(X_test)
print(accuracy_score(y_test, y_pred), ": is the accuracy score")
```

```
0.8245614035087719 : is the accuracy score
```

Screenshot 16

As we can see in the above screenshots 14, 16 - the accuracy score of the test data is same for Naïve Bayes model using grid search cv and without using grid search cv method.

KNN Model

```
X_train_scaled=ss.fit_transform(X_train) #scaling the data since KNN is a distance based algorithm.
X_test_scaled=ss.transform(X_test)
KNN_model=KNeighborsClassifier()
KNN_model.fit(X_train_scaled,y_train)
```

```
KNeighborsClassifier()
```

Screenshot 17

Train Data Accuracy Score

```
0.8548539114043355
```

Screenshot 18

Test Data Accuracy Score

```
0.8245614035087719
```

Screenshot 19

Grid Search CV on KNN Model

```
k_range = list(range(1, 31))
param_grid = dict(n_neighbors=k_range)
```

```
defining parameter range
rid = GridSearchCV(KNN_model, param_grid, cv=10, scoring='accuracy', return_train_score=False,verbose=1

fitting the model for grid search
rid_search=grid.fit(X_test_scaled, y_test)
```

```
Fitting 10 folds for each of 30 candidates, totalling 300 fits
```

```
print(grid_search.best_params_)
```

```
accuracy = grid_search.best_score_ *100

print("Accuracy is : {:.2f}%".format(accuracy))
```

```
Accuracy is : 83.55%
```

Screenshot 20

As we can see in the above screenshots 19, 20- the accuracy score of the test data is almost similar for KNN model using grid search cv and without using grid search cv method.

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

Model tuning is the process of maximizing the model's performance without overfitting or creating too high variance. In machine learning this is accomplished by selecting appropriate hyper-parameters.

Grid search cv method is one of the most common methods of optimizing the parameters. As we have performed grid search cv method in LDA, Logistic regression, KNN, Naïve Bayes, let's check for Bagging and Boosting also.

Models such as Bagging, Boosting, etc are prone to under fitting or over fitting of the data.

Over fitting means, the model works very well on the train data but works relatively poor in the test data.

Under fitting means, the model works very well on the test data but works relatively poor on the training data.

Bagging

Bagging is an ensemble technique. Ensemble techniques are the machine learning techniques that combine several base models to get an optimal model.

Bagging is designed to improve the performance of existing machine learning algorithms used in statistical classification or regression. It is most used with tree-based algorithms.it is a parallel method.

```
RF = RandomForestClassifier()
Bagging_model=BaggingClassifier(base_estimator=RF,n_estimators=100,random_state=1)
Bagging_model.fit(X_train, y_train)
```

```
BaggingClassifier(base_estimator=RandomForestClassifier(), n_estimators=100,
                  random_state=1)
```

Screenshot 21

Train Data Accuracy Score

```
0.9679547596606974
```

Screenshot 22

Test Data Accuracy Score

```
0.8289473684210527
```

Screenshot 23

Grid Search CV on Bagging

```
GridSearchCV(cv=10,
             estimator=BaggingClassifier(base_estimator=RandomForestClassifier(),
                                         max_features=0.5, n_estimators=100),
             param_grid={'base_estimator__max_depth': [3, 4, 5],
                         'max_samples': [0.1, 0.2, 0.5]})
```

```
{'base_estimator__max_depth': 5, 'max_samples': 0.5}
```

```
BaggingClassifier(base_estimator=RandomForestClassifier(max_depth=5),
                  max_features=0.5, max_samples=0.5, n_estimators=100)
```

Screenshot 24

Bagging Accuracy score for the test data

```
0.8048245614035088
```

Screenshot 25

As we can see in the above tables- the accuracy score of test data is almost similar for Bagging model using grid search cv and without using grid search cv method.

Boosting

Boosting is also an ensemble technique. It converts weak learners to strong learners. Unlike bagging it is sequential method where result from one weak learner becomes the input for the another and so on, thus improving the performance of the model.

Each time base learning algorithm is applied, it generates a new weak learner prediction rule. This is an iterative process, and the boosting algorithm combines these weak rules into single strong prediction rule.

There are many kinds of boosting techniques, and the following techniques are used for this given Election dataset.

1. Ada Boosting

2. Gradient Boosting

Ada Boosting

This model is used to increase the efficiency of both binary classifiers and multiclass classifiers as well.

AdaBoost can be applied on top of any classifier method to learn from its issues and bring about more accurate model and thus it is called “best ot-of-the-box classifier”.

```
ADB_model = AdaBoostClassifier(n_estimators=100,random_state=1)
ADB_model.fit(X_train,y_train)
```

```
AdaBoostClassifier(n_estimators=100, random_state=1)
```

Screenshot 26

Train Data Accuracy Score

```
0.8501413760603205
```

Screenshot 27

Test Data Accuracy Score

```
0.8135964912280702
```

Screenshot 28

Grid Search CV on Ada Boosting

```
ada=AdaBoostClassifier()
search_grid={'n_estimators':[500,1000,2000],'learning_rate':[.001,0.01,.1]}
search=GridSearchCV(estimator=ada,param_grid=search_grid,scoring='accuracy',n_jobs=1)
```

```
search.fit(X,y)
search.best_params_
```

```
{'learning_rate': 0.01, 'n_estimators': 2000}
```

Screenshot 29

Ada Boosting Accuracy score for the test data

```
0.875
```

Screenshot 30

From the above screenshots 28, 30 - there is slight variance in the test score accuracy for Ada Boosting using grid search cv and without using grid search cv.

Model performs better when applied “hyper-parameters” using grid search.

Gradient Boosting

This model is just like Ada Boosting , works by sequentially adding the misidentified predictors and underfitted predictions to the ensemble, ensuring the errors identified previously are corrected.

The major difference lies in what it does with the misidentified value of the previous weak learner. This method tries to fit the new predictor to the residual errors made by the previous one.

```
gbcl = GradientBoostingClassifier(random_state=1)
gbcl = gbcl.fit(X_train, y_train)
```

Screenshot 31

Train Data Accuracy Score

```
0.8925541941564562
```

Screenshot 32

Test Data Accuracy Score

```
0.8355263157894737
```

Screenshot 33

Grid Search CV on Gradient Boosting

```
GB = GradientBoostingClassifier(learning_rate=0.1, n_estimators=100,max_depth=3, min_samples_split=2, m
GB.fit(X_train,y_train)
predictors=list(X_train)
print('Accuracy of the GBM : {:.3f}'.format(GB.score(X_test, y_test)))
```

```
Accuracy of the GBM : 0.840
```

Screenshot 34

As we can see in the above screenshots 33, 34 - the accuracy score of the test data is almost similar for Gradient Boosting using grid search cv and without using grid search cv method.

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.

Logistic Regression Model

Classification report on the Train data

	precision	recall	f1-score	support
0	0.75	0.64	0.69	307
1	0.86	0.91	0.89	754
accuracy			0.83	1061
macro avg	0.81	0.78	0.79	1061
weighted avg	0.83	0.83	0.83	1061

Table 15: Classification report – Logistic Regression train data

Classification report on the Test data

	precision	recall	f1-score	support
0	0.76	0.73	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

Table 16: Classification report – Logistic Regression test data

Confusion matrix on the Train data

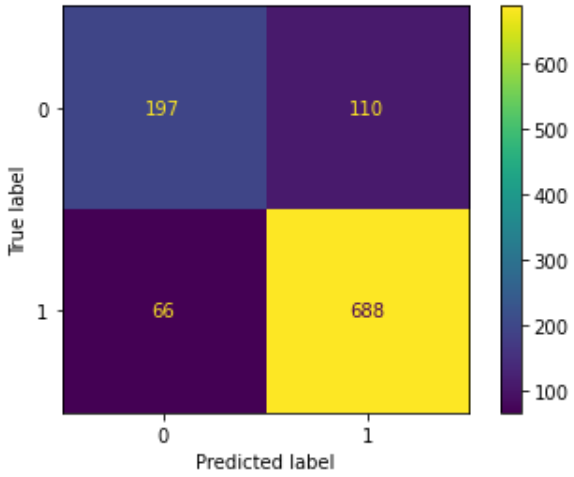

Figure 24: Confusion matrix – Logistic Regression train data

Confusion matrix on the Test data

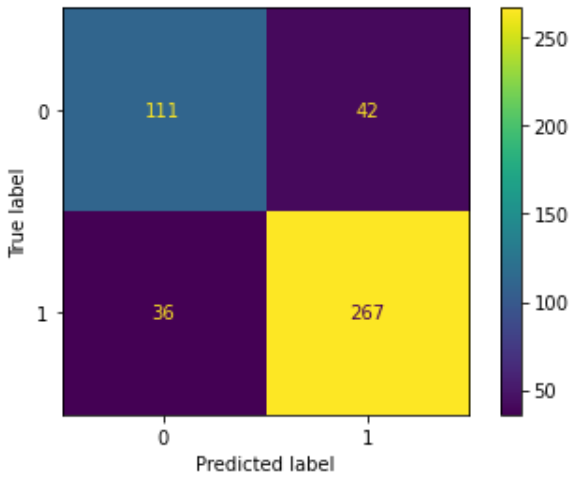

Figure 25: Confusion matrix – Logistic Regression test data

AUC and ROC for the Train data

AUC: 0.890

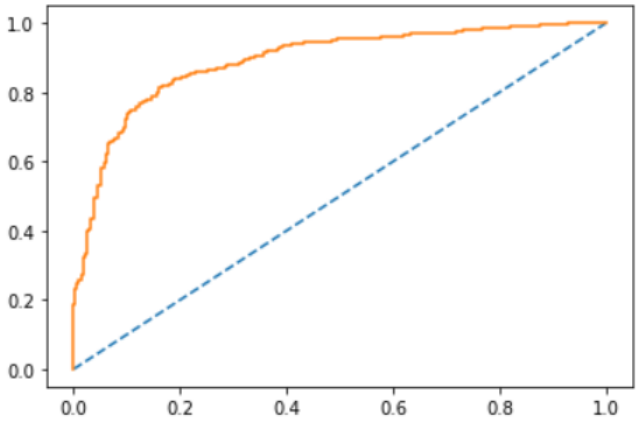

Figure 26: ACU, ROC – Logistic Regression train data

AUC and ROC for the Test data

AUC: 0.890

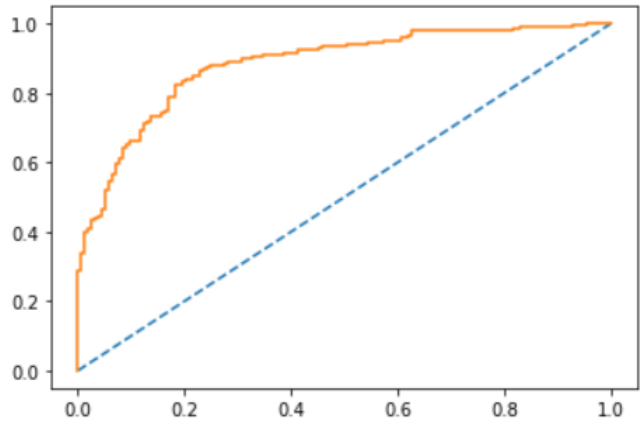

Figure 27: ACU, ROC – Logistic Regression test data

Linear Discriminant Analysis Model

Classification report on the Train data

	precision	recall	f1-score	support
0	0.74	0.65	0.69	307
1	0.86	0.91	0.89	754
accuracy			0.83	1061
macro avg	0.80	0.78	0.79	1061
weighted avg	0.83	0.83	0.83	1061

Table 17: Classification report – LDA train data

Classification report on the Test data

	precision	recall	f1-score	support
0	0.76	0.73	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

Table 18: Classification report – LDA test data

Confusion matrix on the Train data

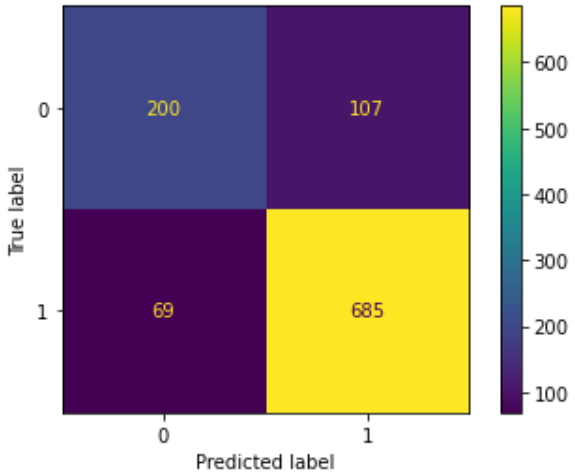

Figure 28: Confusion matrix – LDA train data

Confusion matrix on the Test data

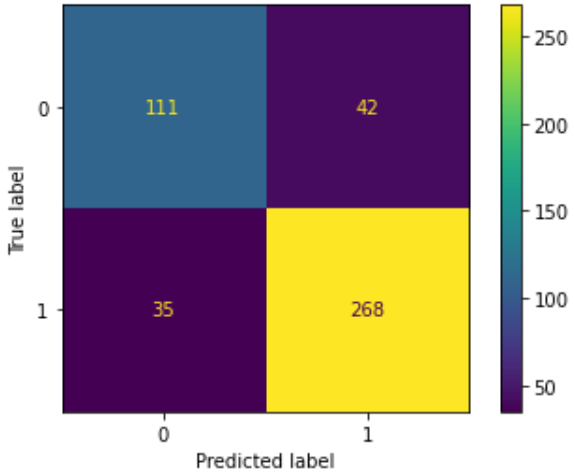

Figure 29: Confusion matrix – LDA test data

AUC and ROC for the Train data

AUC: 0.890

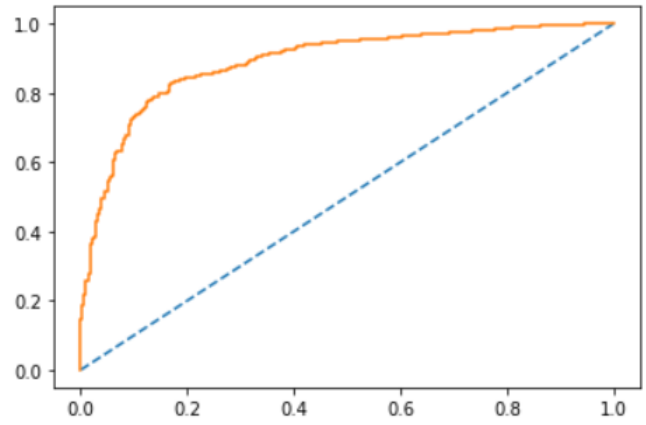

Figure 30: ACU, ROC – LDA train data

AUC and ROC for the Test data

AUC: 0.888

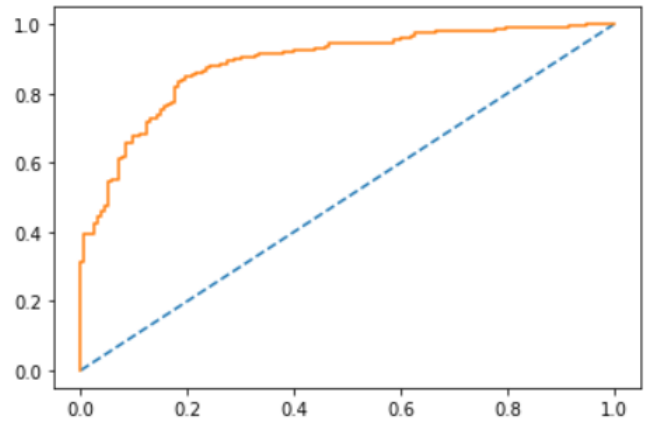

Figure 31: ACU, ROC – LDA test data

Naïve Bayes Model

Classification report on the Train data

	precision	recall	f1-score	support
0	0.72	0.69	0.71	307
1	0.88	0.89	0.88	754
accuracy			0.83	1061
macro avg	0.80	0.79	0.80	1061
weighted avg	0.83	0.83	0.83	1061

Table 19: Classification report – Naïve Bayes train data

Classification report on the Test data

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

Table 20: Classification report – Naïve Bayes test data

Confusion matrix on the Train data

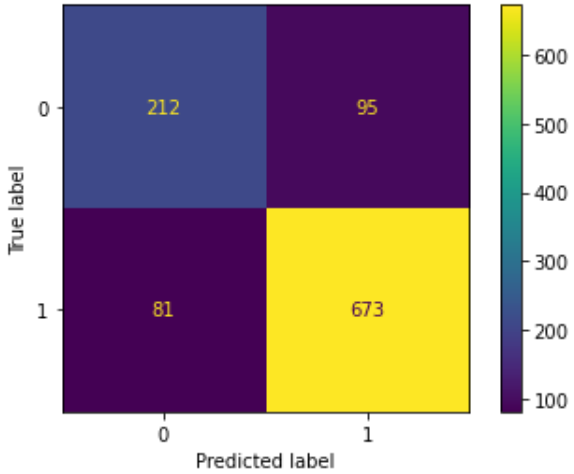

Figure 32: Confusion matrix – Naïve Bayes train data

Confusion matrix on the Test data

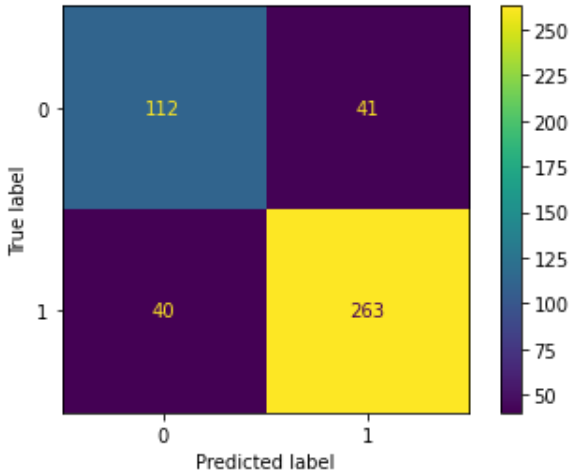

Figure 33: Confusion matrix – Naïve Bayes test data

AUC and ROC for the Train data

AUC: 0.889

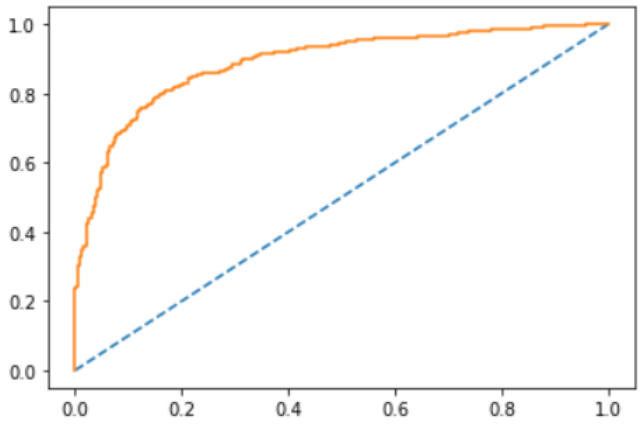

Figure 34: ACU, ROC – Naïve Bayes train data

AUC and ROC for the Test data

AUC: 0.876

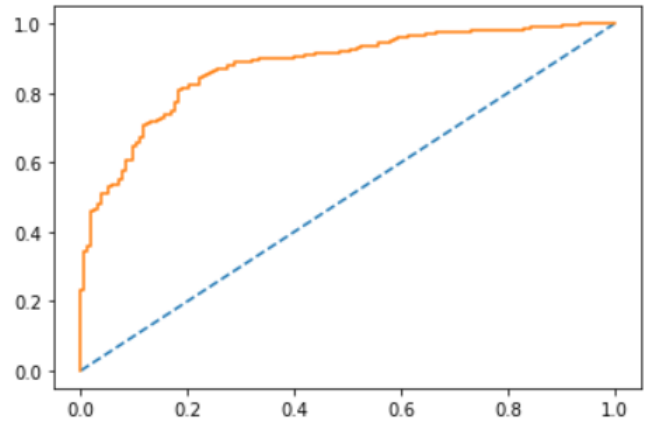

Figure 35: ACU, ROC – Naïve Bayes test data

KNN Model

Classification report on the Train data

	precision	recall	f1-score	support
0	0.77	0.70	0.74	307
1	0.88	0.92	0.90	754
accuracy			0.85	1061
macro avg	0.83	0.81	0.82	1061
weighted avg	0.85	0.85	0.85	1061

Table 21: Classification report – KNN train data

Classification report on the Test data

	precision	recall	f1-score	support
0	0.75	0.71	0.73	153
1	0.86	0.88	0.87	303
accuracy			0.82	456
macro avg	0.81	0.80	0.80	456
weighted avg	0.82	0.82	0.82	456

Table 22: Classification report – KNN test data

Confusion matrix on the Train data

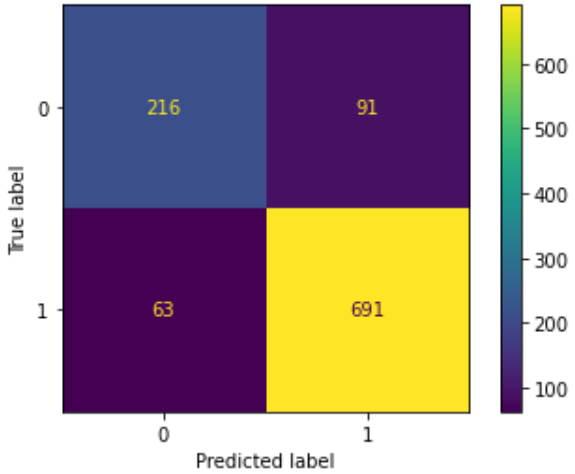

Figure 36: Confusion matrix – KNN train data

Confusion matrix on the Test data

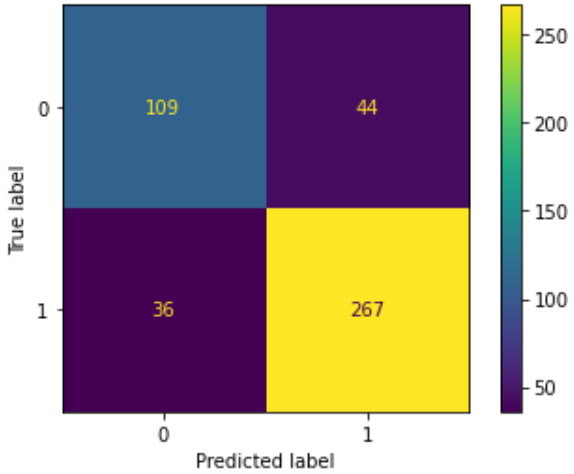

Figure 37: Confusion matrix – KNN test data

AUC and ROC for the Train data

AUC: 0.929

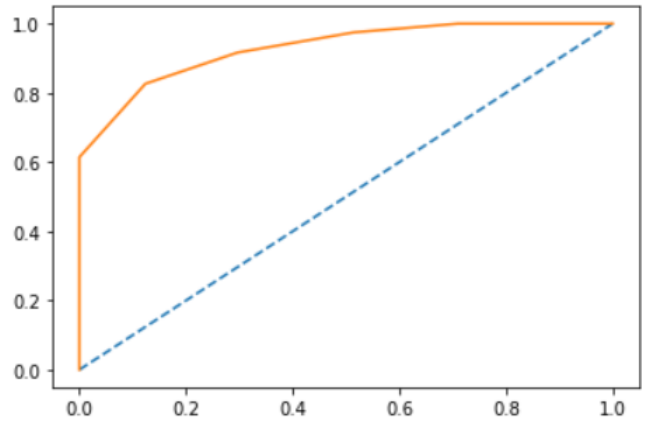

Figure 38: ACU, ROC – KNN train data

AUC and ROC for the Test data

AUC: 0.866

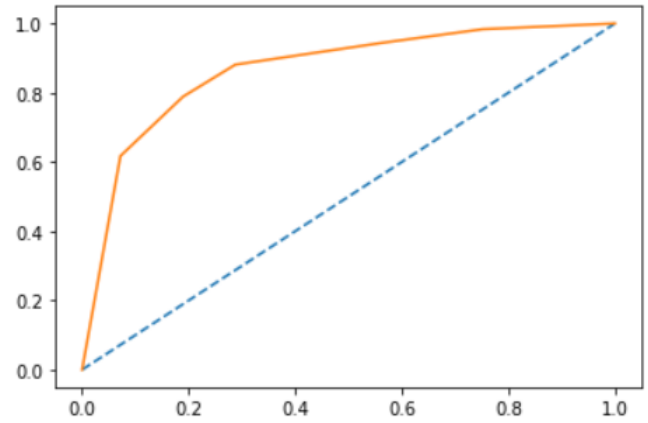

Figure 39: ACU, ROC – KNN test data

Bagging

Classification report on the Train data

	precision	recall	f1-score	support
0	0.99	0.90	0.94	307
1	0.96	0.99	0.98	754
accuracy			0.97	1061
macro avg	0.97	0.95	0.96	1061
weighted avg	0.97	0.97	0.97	1061

Table 23: Classification report – Bagging train data

Classification report on the Test data

	precision	recall	f1-score	support
0	0.78	0.68	0.73	153
1	0.85	0.90	0.88	303
accuracy			0.83	456
macro avg	0.82	0.79	0.80	456
weighted avg	0.83	0.83	0.83	456

Table 24: Classification report – Bagging test data

Confusion matrix on the Train data

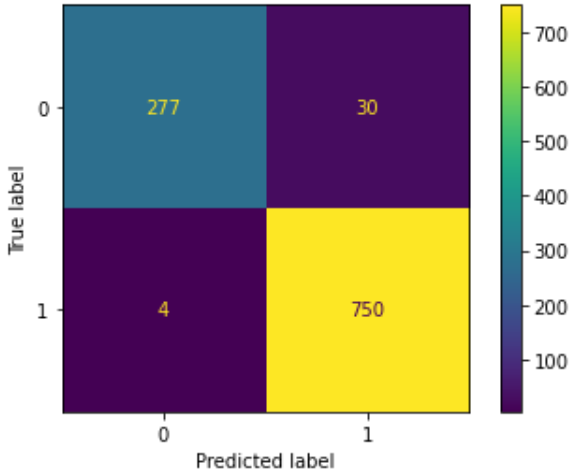

Figure 40: Confusion matrix – Bagging train data

Confusion matrix on the Test data

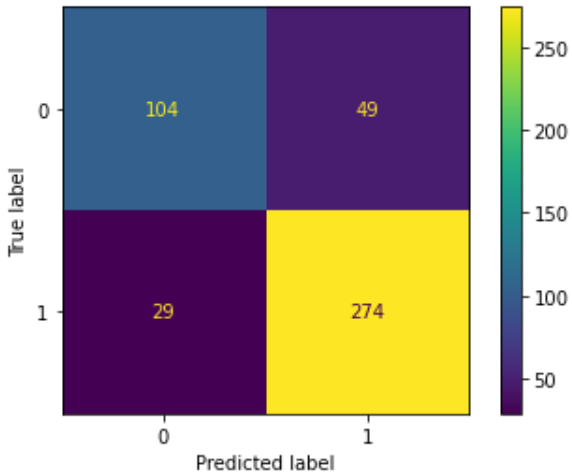

Figure 41: Confusion matrix – Bagging test data

AUC and ROC for the Train data

AUC: 0.997

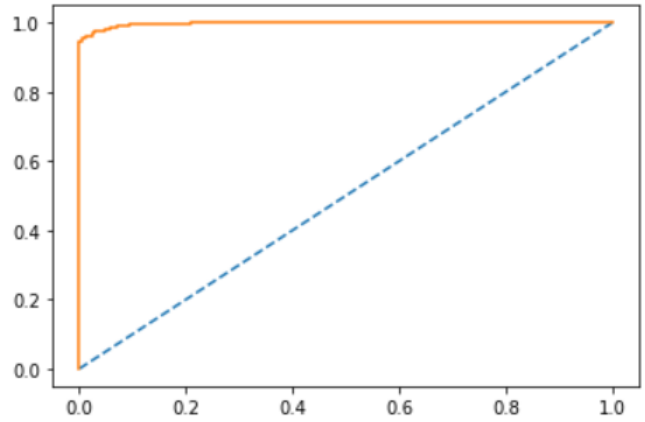

Figure 42: ACU, ROC – Bagging train data

AUC and ROC for the Test data

AUC: 0.897

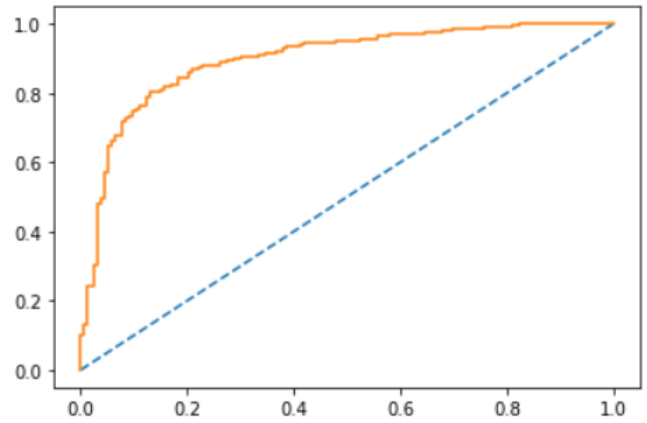

Figure 43: ACU, ROC – Bagging test data

Ada Boosting

Classification report on the Train data

	precision	recall	f1-score	support
0	0.76	0.70	0.73	307
1	0.88	0.91	0.90	754
accuracy			0.85	1061
macro avg	0.82	0.80	0.81	1061
weighted avg	0.85	0.85	0.85	1061

Table 25: Classification report – Ada Boosting train data

Classification report on the Test data

	precision	recall	f1-score	support
0	0.75	0.67	0.71	153
1	0.84	0.88	0.86	303
accuracy			0.81	456
macro avg	0.79	0.78	0.79	456
weighted avg	0.81	0.81	0.81	456

Table 26: Classification report – Ada Boosting test data

Confusion matrix on the Train data

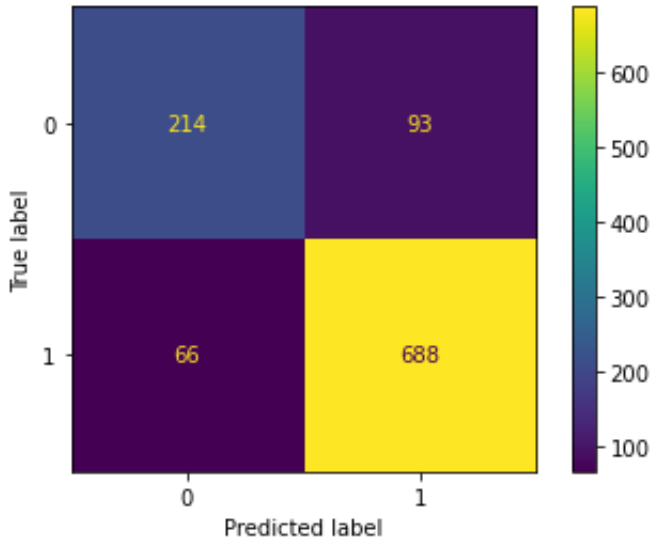

Figure 44: Confusion matrix – Ada Boosting train data

Confusion matrix on the Test data

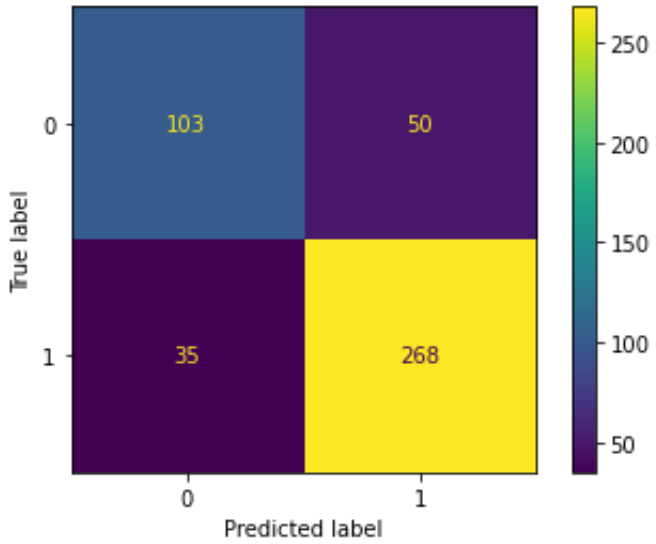

Figure 45: Confusion matrix – Ada Boosting test data

AUC and ROC for the Train data

AUC: 0.915

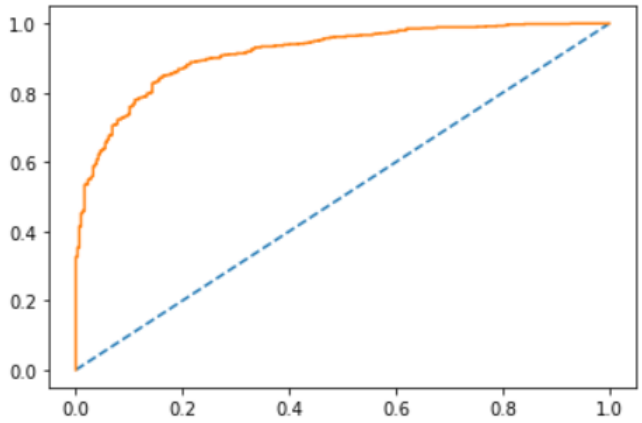

Figure 46: ACU, ROC – Ada Boosting train data

AUC and ROC for the Test data

AUC: 0.877

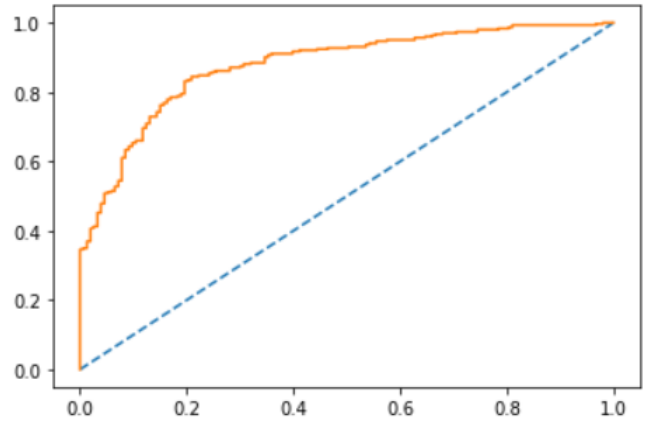

Figure 47: ACU, ROC – Ada Boosting test data

Gradient Boosting

Classification report on the Train data

	precision	recall	f1-score	support
0	0.84	0.78	0.81	307
1	0.91	0.94	0.93	754
accuracy			0.89	1061
macro avg	0.88	0.86	0.87	1061
weighted avg	0.89	0.89	0.89	1061

Table 27: Classification report – Gradient Boosting train data

Classification report on the Test data

	precision	recall	f1-score	support
0	0.80	0.69	0.74	153
1	0.85	0.91	0.88	303
accuracy			0.84	456
macro avg	0.82	0.80	0.81	456
weighted avg	0.83	0.84	0.83	456

Table 28: Classification report – Gradient Boosting test data

Confusion matrix on the Train data

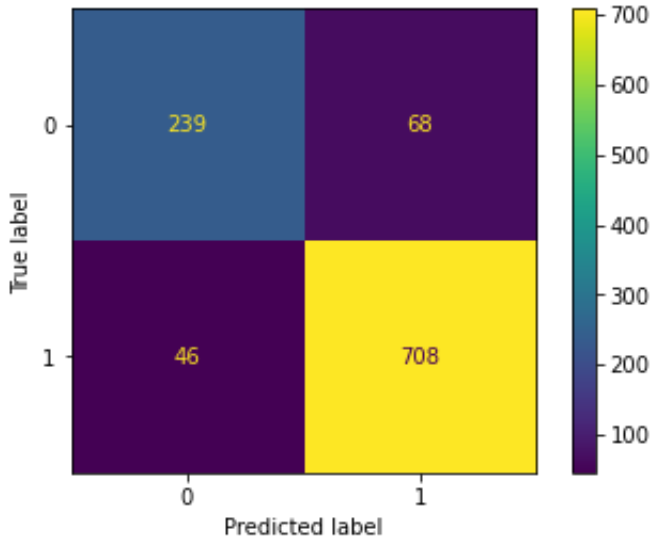

Figure 48: Confusion matrix – Gradient Boosting train data

Confusion matrix on the Test data

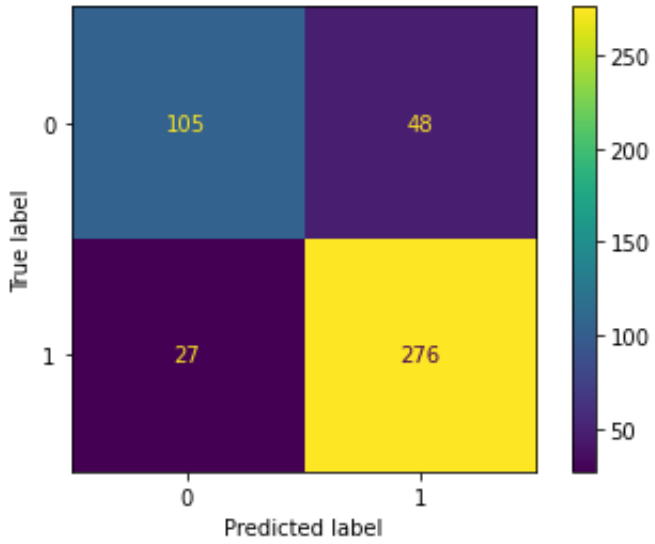

Figure 49: Confusion matrix – Gradient Boosting test data

AUC and ROC for the Train data

AUC: 0.951

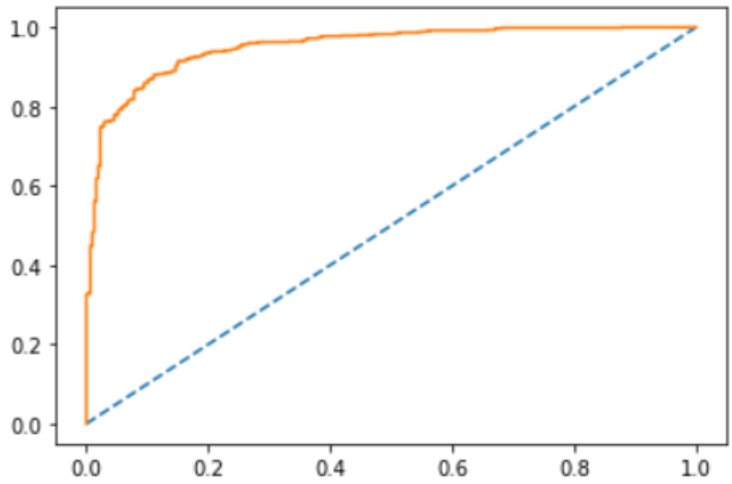

Figure 50: ACU, ROC – Gradient Boosting train data

AUC and ROC for the Test data

AUC: 0.899

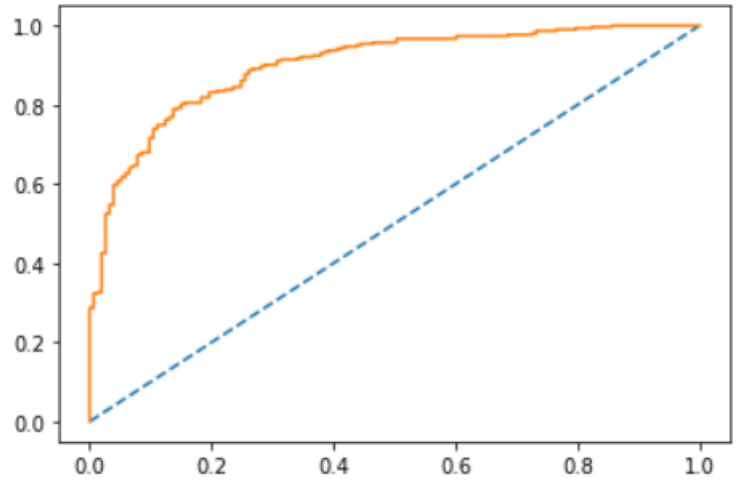

Figure 51: ACU, ROC – Gradient Boosting test data

Inferences:

Below are the model's metrics - comparison tables.

Metrics	LR Train	LR Test	LDA Train	LDA Test
Accuracy	83.41	82.8	83.41	83.11
AUC	89.0	89.0	89.0	88.8
Recall	91.0	88.0	91.0	88.0
Precision	86.0	86.0	86.0	86.0
F1 Score	89.0	89.0	89.0	87.0

Table 29: Metrics comparison – Logistic Regression, LDA

LR – Logistic Regression

LDA – Linear Discriminant Analysis

Metrics	KNN Train	KNN Test	NB Train	NB Test
Accuracy	85.48	82.45	83.41	82.23
AUC	92.9	86.6	88.9	87.6
Recall	92.0	88.0	89.0	87.0
Precision	88.0	86.0	88.0	87.0
F1 Score	90.0	87.0	88.0	87.0

Table 30: Metrics comparison – KNN, Naïve Bayes

KNN – K Nearest Neighbour

NB – Naïve Bayes

Metrics	Bagging Train	Bagging Test	AdBoost Train	AdBoost Test	GdBoost Train	GdBoost Test
Accuracy	96.79	82.89	85.01	81.35	89.25	83.55
AUC	99.7	89.7	91.5	87.7	95.1	89.9
Recall	99.0	90.0	91.0	88.0	94.0	91.0
Precision	96.0	85.0	88.0	84.0	91.0	85.0
F1 Score	98.0	88.0	90.0	86.0	93.0	88.0

Table 31: Metrics comparison – Bagging, Ada Boosting, Gradient Boosting

AdBoost – Ada Boosting

GdBoost – Gradient Boosting

Insights

1. From the above table-no, when compared to Accuracy scores of both models LR and LDA scores remains same with slight variation, where LDA model performs better than Logistic Regression.

   Since problem is on predictions, when we check for precision scores for LR and LDA models, there is no difference. Predictions are same.

2. From the above table-no, when compared to Accuracy scores of both models KNN and NB there is slight variation in the score, where NB model performs better than KNN.

   Since problem is on predictions, when we check for precision scores, NB model performs better compared to KNN.

3. From the above table-no, when compared to Accuracy scores among 3 models – Bagging, Ada Boosting and Gradient Boosting, Ada Boosting model performs better with slight variance in accuracy score.

   Since problem is on predictions, when we check for precision scores among 3 models – Bagging, Ada Boosting and Gradient Boosting, Ada Boosting model performs better with slight variance.

1.8 Based on these predictions, what are the insights?

Our main business objective is to build a model, to predict which party voter will vote based on the given information and to create exit poll that will help in predicting overall win and seats covered by a particular party.

Insights

- When compared to all models by checking all score both Logistic Regression and LDA models performs better with very slight variation.
- Given the target variable “Vote” is categorised into labels – 0,1 where 0 being “Conservative Party” and 1 being “Labour Party”.
- From LR and LDA model’s predictions, Labour party claims a greater number of votes compared to Conservative Party.

Recommendations

- Gathering more information like Constituency, Category, Education, Liabilities, etc can help in training models and thus improving their predictive powers.
- Hyper-parameters tuning is an important aspect of model building. There are limitations to process these combinations where huge amount of processing power is required. But if tuning can be done with many sets of parameters than we might get even better results.
- We can also create a function in which all the models predict the outcome in sequence. This will help in better understanding and the probability of what the outcome will be.

PROBLEM 2

Executive Summary:

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

Data Introduction:

The purpose of this exercise is to explore the data and is recommended for learning and practicing our skills using Text Mining and Sentiment Analysis.

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

Characters

Number of characters in Roosevelt speech are 7571.

Number of characters in Kennedy speech are 7618.

Number of characters in Nixon speech are 9991.

Words

Number of words in Roosevelt speech are 1536.

Number of words in Kennedy speech are 1546.

Number of words in Nixon speech are 2028.

Sentences

Number of sentences in Roosevelt speech are 68.

Number of sentences in Kennedy speech are 52.

Number of sentences in Nixon speech are 69.

2.2 Remove all the stop words from all three speeches.

To remove the stop words, there is a package called “stopwords” in the nltk.corpus library.

So, in order to do so we need to import library “from nltk.corpus import stopwords”.

The stop words library contains following words as shown in the below table.

```
stopwords
```

```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
```

Screenshot 35

Stop words usually don’t have any importance in understanding the sentiment or usefulness in machine learning algorithms. These stop words present in the package are universally accepted stop words and we can add using the (.extend()) function or remove them as per our requirement.

Also, we need to specify the language we are working with before defining the functions, as there are many language packages.

Sample data after removing stop words in Roosevelt speech:

```
['national',
 'day',
 'inauguration',
 'since',
 '1789',
 'people',
 'renewed',
 'sense',
 'dedication',
 'united',
 'states',
 'washington',
 'day',
 'task',
 'people',
 'create',
 'weld',
 'together',
 'nation',
```

Screenshot 36

Sample data after removing stop words in Kennedy speech:

```
['vice',
 'president',
 'johnson',
 'mr',
 'speaker',
 'mr',
 'chief',
 'justice',
 'president',
 'eisenhower',
 'vice',
 'president',
 'nixon',
 'president',
 'truman',
 'reverend',
 'clergy',
 'fellow',
 'citizens',
```

Screenshot 37

Sample data after removing stop words in Nixon speech:

```
['mr',
 'vice',
 'president',
 'mr',
 'speaker',
 'mr',
 'chief',
 'justice',
 'senator',
 'cook',
 'mrs',
 'eisenhower',
 'fellow',
 'citizens',
 'great',
 'good',
 'country',
 'share',
 'together',
```

Screenshot 38

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 stop words)

Results after removing stop words in all 3 speeches are below.

Roosevelt Speech

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

Screenshot 39

Most occurring word: Nation

Kennedy Speech

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

Screenshot 40

Most occurring word: Let

Nixon Speech

```
[('us', 26), ('let', 22), ('america', 21),
```

Screenshot 41

Most occurring word: Us

2.4 Plot the word cloud of each of the speeches of the variable. (After removing the stop words)

Word Cloud is a data visualization technique used for representing test data in which the size of each word indicates its frequency or importance.

Word Cloud of Roosevelt's Speech

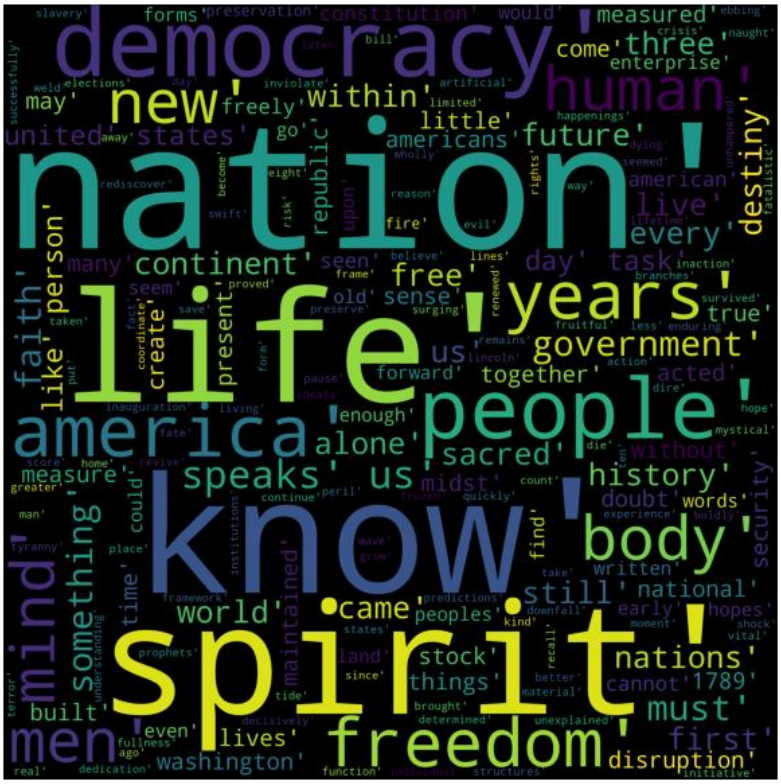

Figure 52: Word cloud – Roosevelt's speech

We can see some highlighted words like "nation", "know", "life", "spirit" which are bigger in size and most frequently used or important words.

Word Cloud of Kennedy's Speech

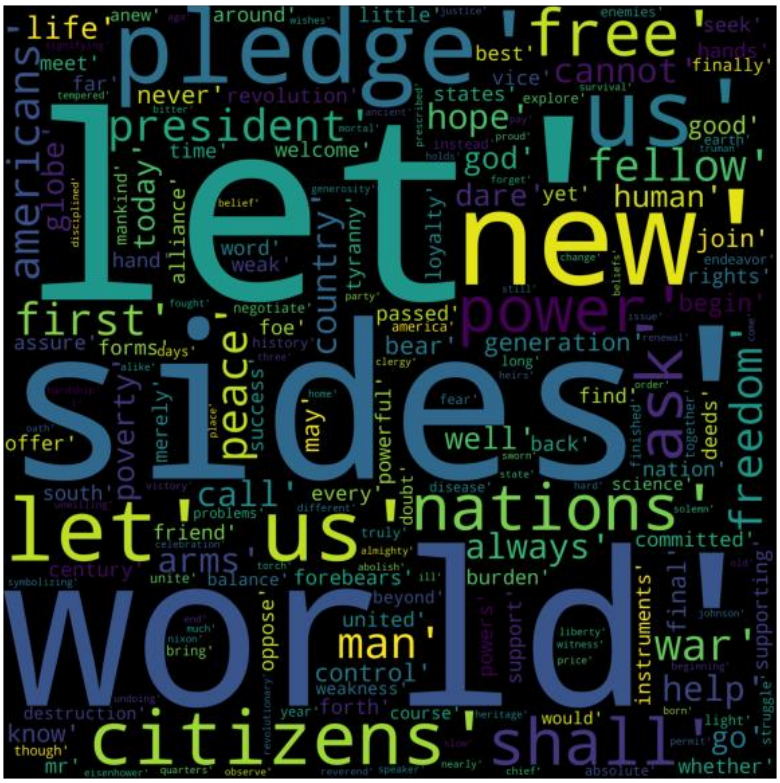

Figure 53: Word cloud – Kennedy's speech

We can see some highlighted words like "let", "sides", "world" which are bigger in size and most frequently used or important words.

Word Cloud of Nixon's Speech

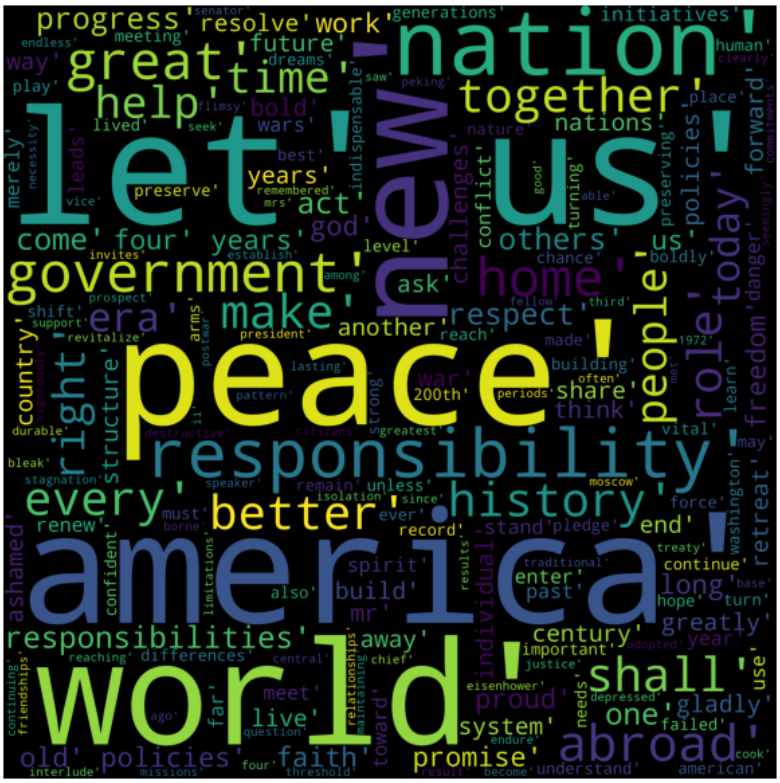

Figure 54: Word cloud – Nixon's speech

We can see some highlighted words like "let", "us", "america", "world" which are bigger in size and most frequently used or important words.

Insights

- Our objective is to look at all 3 speeches and analyze them to find the strength and sentiment of the speeches.
- Based on the outputs we can see that there are some similar words present is all the speeches.
- These words may be inspired by many people and helped them to won as the President of United States of America.

Among all the 3 speeches "nation" is the word that is significantly highlighted