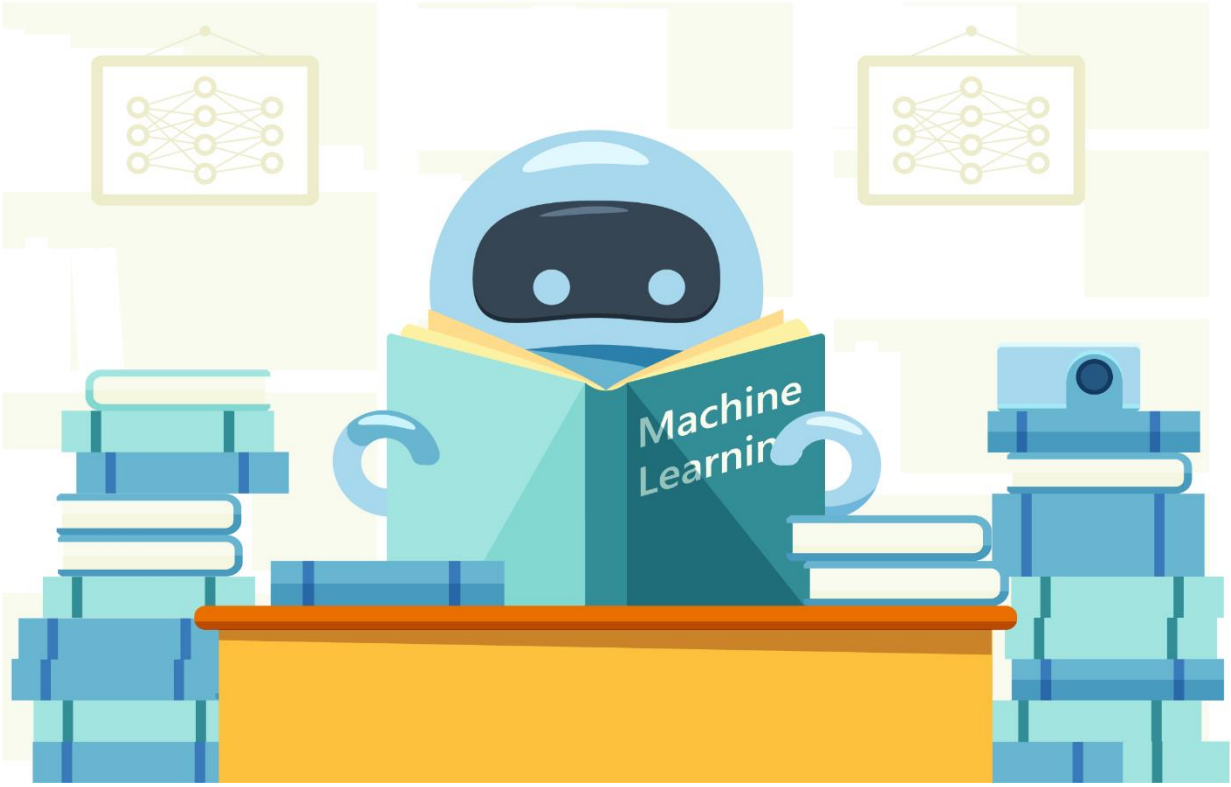
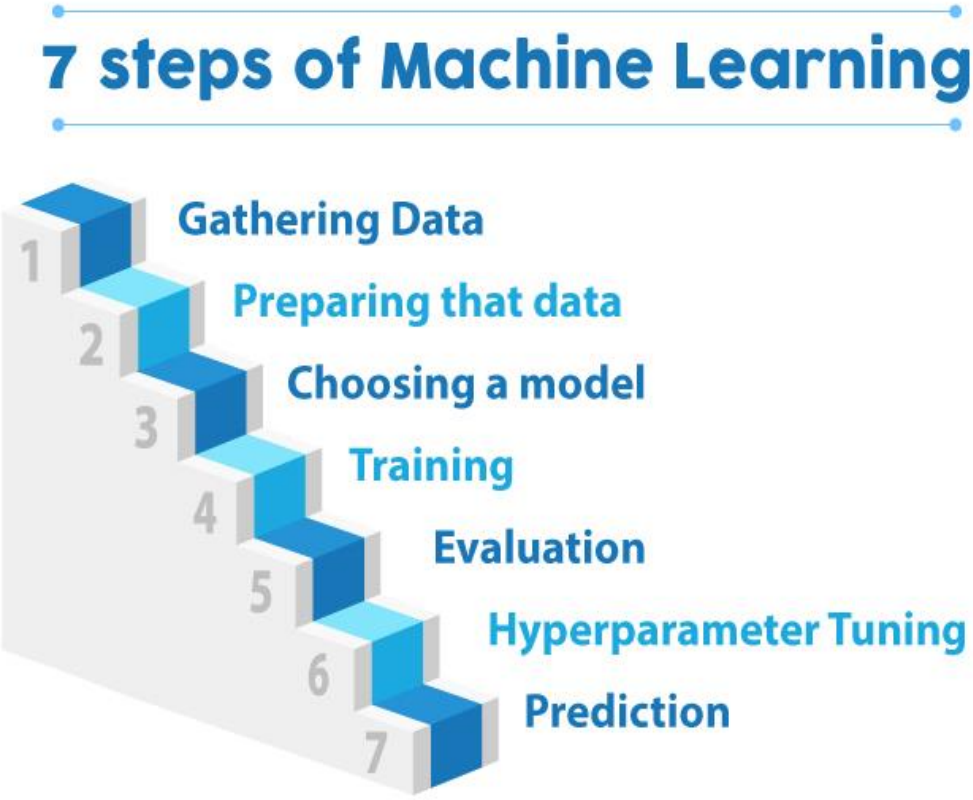
PARAKANDLA. MANISHA (PGP-DSBA GREATLEARNING ONLINE)

23-01-2022 – MACHINE LEARNING PROJECT





Contents

List of Tables:

List of Figures:

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 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.

Dataset for Problem: Election_Data.xlsx

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

The dataset is having the survey data of 1525 voters with 9 features.

SAMPLE OF THE DATASET:

	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
...	...	...	...	...	...	...	...	...	...	...
1520	1521	Conservative	67	5	3	2	4	11	3	male
1521	1522	Conservative	73	2	2	4	4	8	2	male
1522	1523	Labour	37	3	3	5	4	2	2	male
1523	1524	Conservative	61	3	3	1	4	11	2	male
1524	1525	Conservative	74	2	3	2	4	11	0	female

1525 rows × 10 columns

The dataset is having 1525 rows and 10 columns

The unnamed:0 column is nothing but the index. So, I have dropped that column. Now the dataset is having 1525 rows and 9 features as columns.

	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 1.1 sample of Election_Data

DISTRIBUTION SUMMARY OF THE DATASET:

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

Table1.2 Summary of Election_data

Describe () is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values. The describe method is used to get the 5 number summary of the dataset.

Info () prints information about a data frame including the index data type and columns, non-null values and memory usage.

```
<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
```

From the info (), we can observe that there are no null values present in the dataset.

Out of 9 features, 7 are of integer datatype and the other 2 are of object datatype.

Skewness:

Skew () function returns unbiased skew over requested axis Normalized by N-1. Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. Most of the features are negatively skewed except age & Hauge. The skewness of all the features is represented below.

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

Skewness is a measure of the asymmetry of the probability distribution of a real -valued random variable about its mean. Only two variables are positively skewed and the rest are negatively skewed with max skewedness in Blair.

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

EXPLORATORY DATA ANALYSIS:

Univariate Analysis:

→The Categorical variables in dataset are: ['vote', 'gender']
→The numeric variables in dataset are: ['age', 'economic.cond.national', 'economic.cond.household', 'Blair', 'Hague', 'Europe', 'political.knowledge']
→ Target variable value counts:

```
Labour          1063
Conservative     462
Name: vote, dtype: int64
```

Categorical variables:

```
VOTE   2
Conservative     462
Labour          1063
Name: vote, dtype: int64
GENDER   2
male      713
female    812
Name: gender, dtype: int64
```

Numeric variables:

feature: vote
['Labour', 'Conservative']
Categories (2, object): ['Conservative', 'Labour']
[2]
Categories (1, int64): [2]

feature: age
[43, 36, 35, 24, 41, ..., 86, 85, 87, 93, 91]
Length: 70

Categories (70, int64): [24, 25, 26, 27, ..., 90, 91, 92, 93]
[70]
Categories (1, int64): [70]

feature: economic.cond.national
[3, 4, 2, 1, 5]
Categories (5, int64): [1, 2, 3, 4, 5]
[5]
Categories (1, int64): [5]

feature: economic.cond.household
[3, 4, 2, 1, 5]
Categories (5, int64): [1, 2, 3, 4, 5]
[5]
Categories (1, int64): [5]

feature: Blair
[4, 5, 2, 1, 3]
Categories (5, int64): [1, 2, 3, 4, 5]
[5]
Categories (1, int64): [5]

feature: Hague
[1, 4, 2, 5, 3]
Categories (5, int64): [1, 2, 3, 4, 5]
[5]
Categories (1, int64): [5]

feature: Europe
[2, 5, 3, 4, 6, ..., 1, 7, 9, 10, 8]
Length: 11
Categories (11, int64): [1, 2, 3, 4, ..., 8, 9, 10, 11]
[11]
Categories (1, int64): [11]

feature: political.knowledge
[2, 0, 3, 1]
Categories (4, int64): [0, 1, 2, 3]
[4]
Categories (1, int64): [4]

feature: gender
['female', 'male']
Categories (2, object): ['female', 'male']
[2]
Categories (1, int64): [2]

There are 8 rows, having duplicate values. These duplicate values are no longer useful for the model building & does not contain any useful information. So, I choose to drop them.

The shape of the dataset after dropping the duplicates is (1517,9).

Percentage of target variable:

```
Labour          0.69677
Conservative    0.30323
Name: vote, dtype: float64
```

This concludes that 69% of the voters belong to Labour category. And the rest 30% belongs to Consecutive category.

Univariate & Bivariate Analysis:

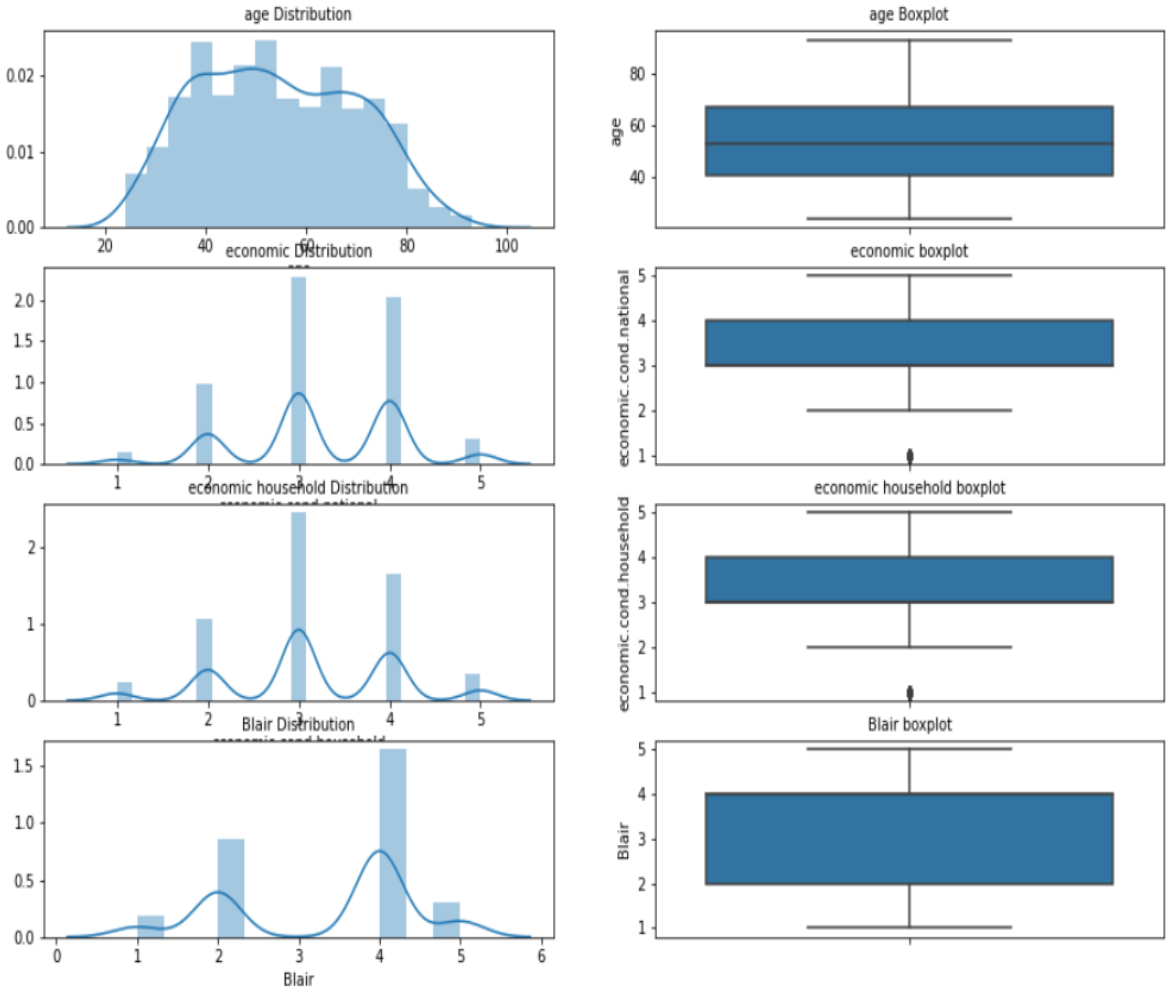


Fig1. Distribution & boxplot—1

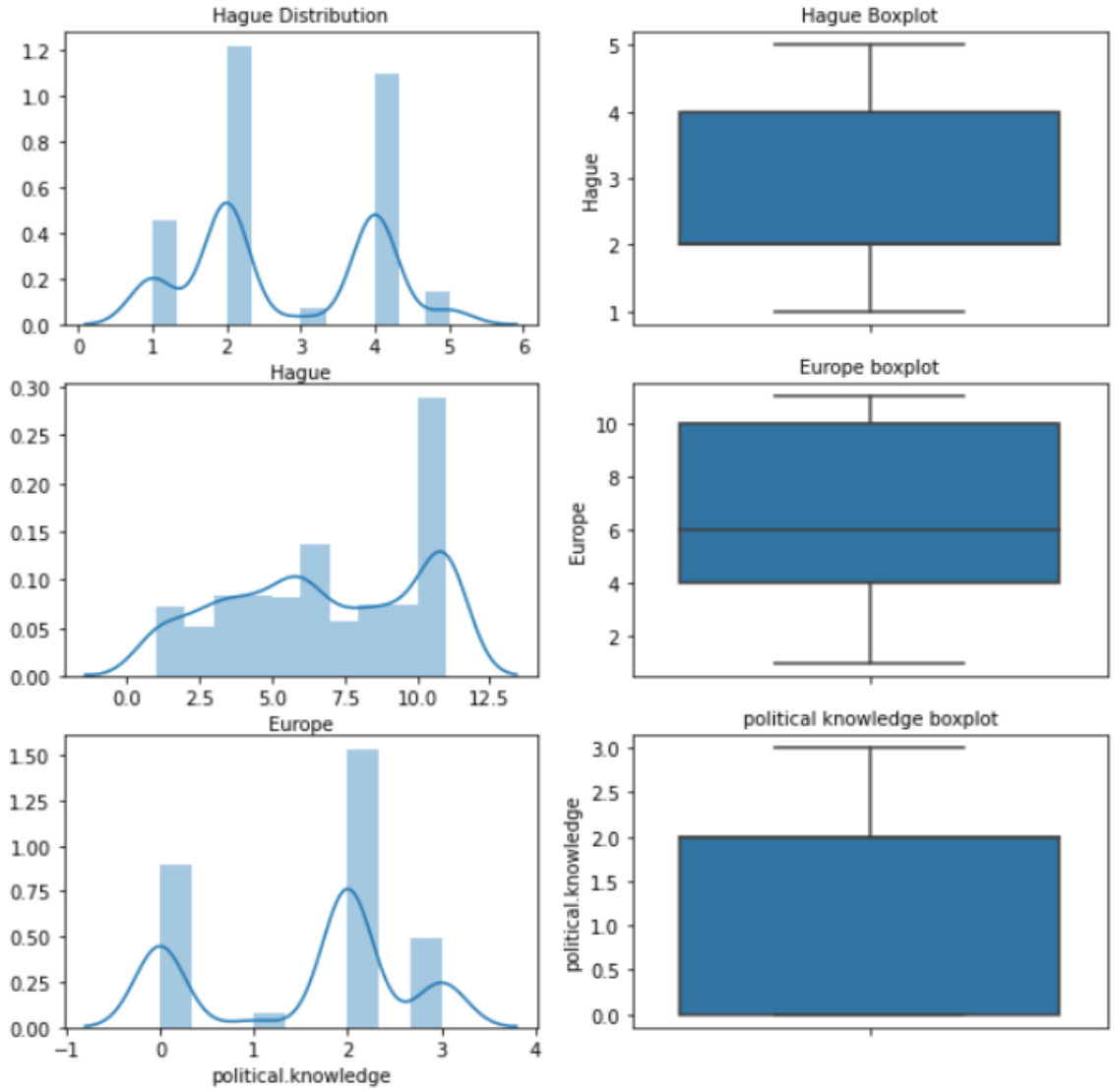


Fig2. Distribution & boxplot—2

We can see that all the numerical variables are normally distributed (not perfectly normal though) and are multi modal in some instances as well. From the boxplots the min and max values of the variables are not very clear, we can separately obtain them while checking for outliers.

Outliers are present in very less amount and that too only in two features, which can be negligible. The distances of the outliers are also less. The proportion of the outliers will be discussed below.

The features having outliers are economic.cond.national & economic.cond.household .

The proportion of outliers in economic.cond.national is 2.4390243902439024
The proportion of outliers in economic.cond.household is 4.284772577455504

The outliers are in very less proportion and many not effect the data. So, I choose not to treat the outliers for this problem.

Bivariate Analysis:

The count plot below represents that most of the voters are belonged to the category labour compared to the category consecutive.

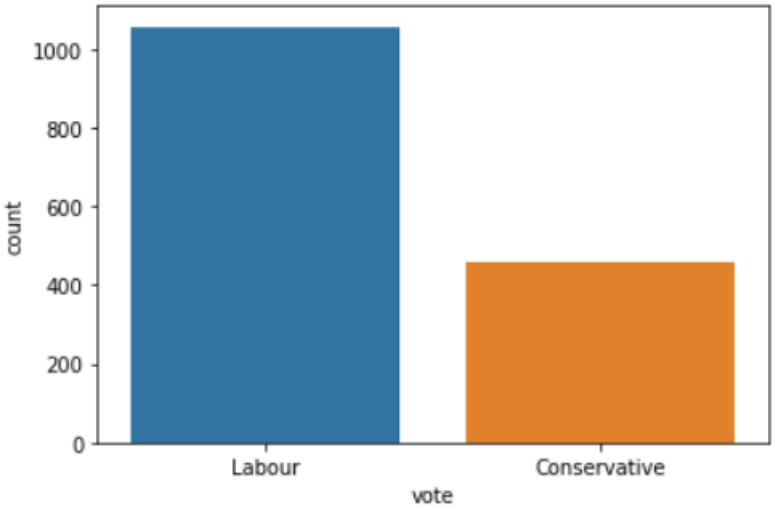
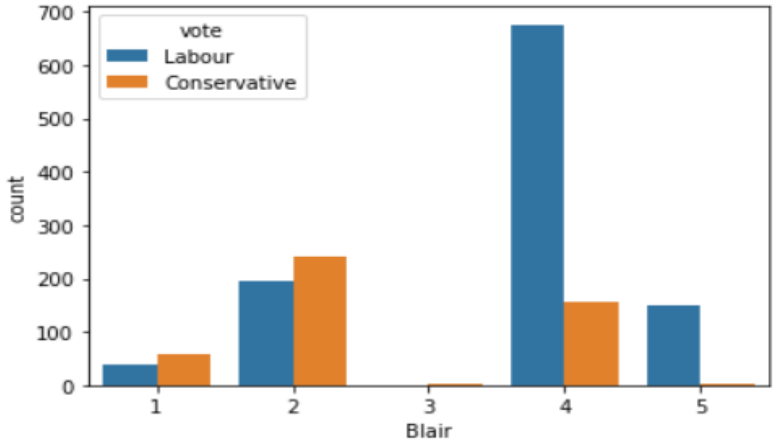


Fig3. Count plot for vote

Count plots with X-axis Blair, Hague and hue as vote:



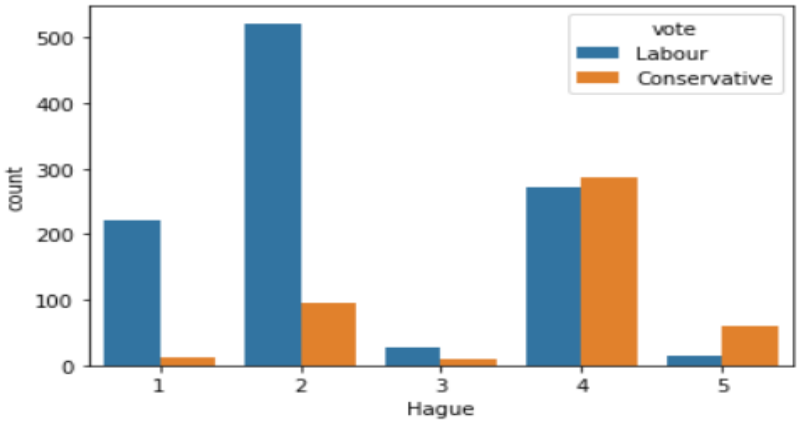


Fig4 Count plots for Blair & Hague

Count plot showing the voters with respect to gender. i.e., gender as x=axis & hue= vote.

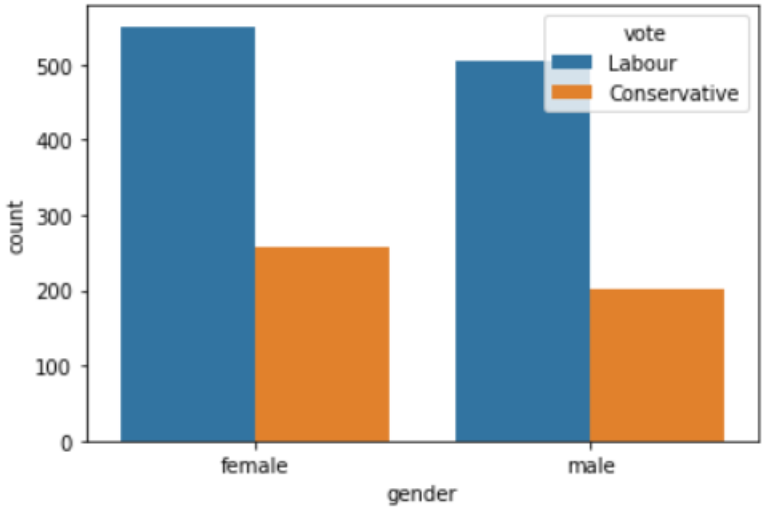


Fig5. Count plot for gender

The shows that there are more female voters who belongs to category labour compared to male.

Count plot representing the voters in Europe. I.e., x =vote, hue = Europe

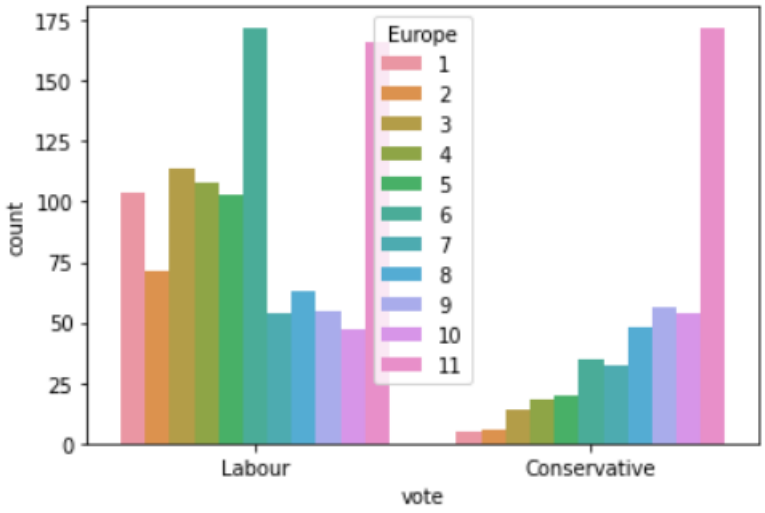


Fig6. Count plot plot for voters in Europe

Multivariate Analysis:

Multivariate analysis can be represented with the pair plot and correlation matrix.

From the pairplot we can see the distribution of all the variables with respect to one another. And, from correlation matrix we can find the highly correlated features with respect to each other.

Pairplot:

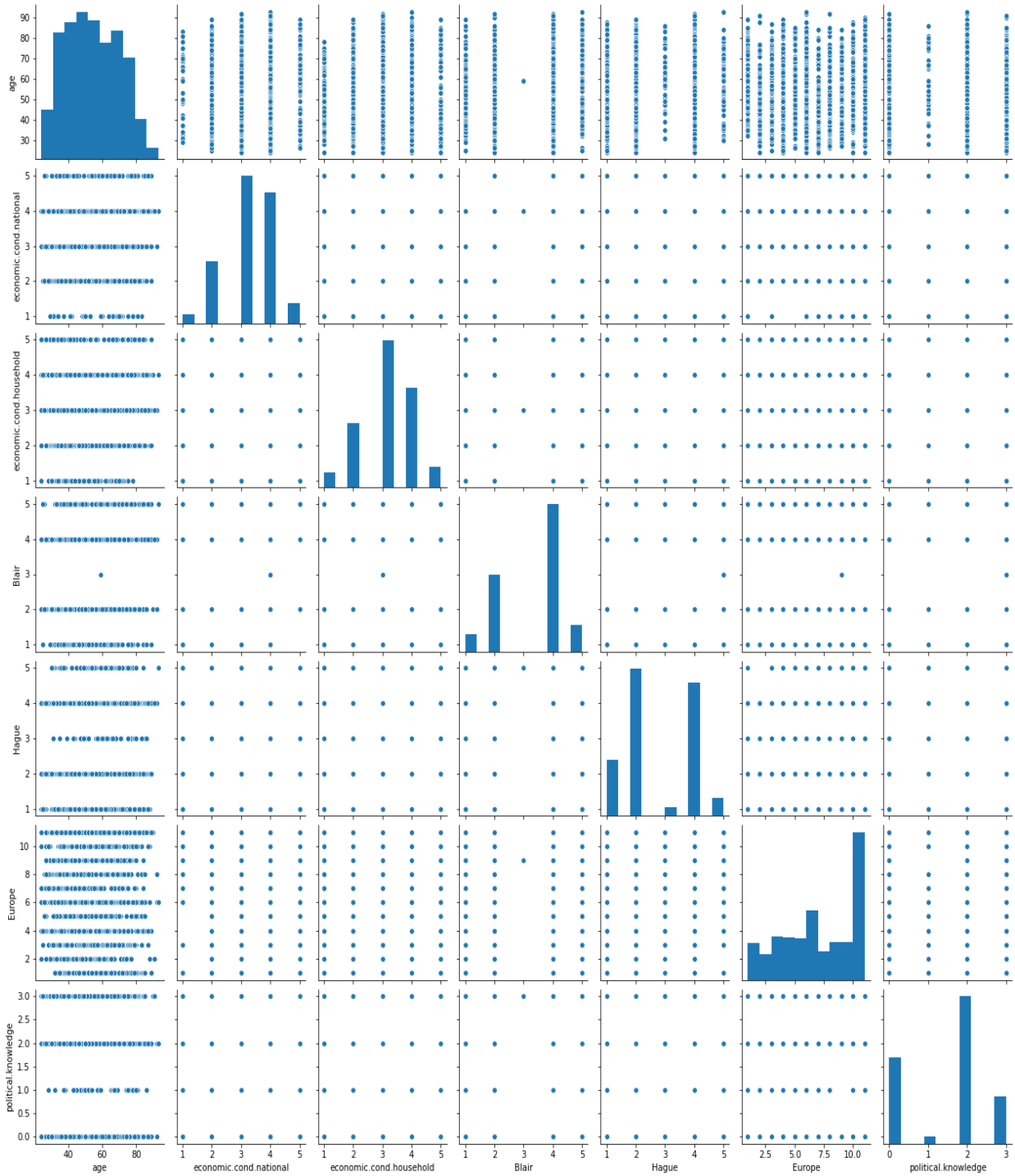


Fig7.Pairplot

Pairplot tells us about the interaction of each variable with every other variable present.

As such there is no strong relationship present between the variables. There is a mixture of positive and negative relationships though which is expected.

Overall, it's a rough estimate of the interactions, clearer picture can be obtained by the heatmap.

Covariance Matrix:

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge
age	246.544655	0.258740	-0.568222	0.591818	0.602692	3.344366	-0.793429
economic.cond.national	0.258740	0.777558	0.285454	0.337851	-0.218216	-0.608432	-0.022481
economic.cond.household	-0.568222	0.285454	0.866890	0.236065	-0.115202	-0.346780	-0.038900
Blair	0.591818	0.337851	0.236065	1.380089	-0.352571	-1.146966	-0.027134
Hague	0.602692	-0.218216	-0.115202	-0.352571	1.519005	1.161811	-0.039970
Europe	3.344366	-0.608432	-0.346780	-1.146966	1.161811	10.883687	-0.540915
political.knowledge	-0.793429	-0.022481	-0.038900	-0.027134	-0.039970	-0.540915	1.175961

Table 1.3 Covariance matrix

Correlation Matrix:

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge
age	1.000000	0.018687	-0.038868	0.032084	0.031144	0.064562	-0.046598
economic.cond.national	0.018687	1.000000	0.347687	0.326141	-0.200790	-0.209150	-0.023510
economic.cond.household	-0.038868	0.347687	1.000000	0.215822	-0.100392	-0.112897	-0.038528
Blair	0.032084	0.326141	0.215822	1.000000	-0.243508	-0.295944	-0.021299
Hague	0.031144	-0.200790	-0.100392	-0.243508	1.000000	0.285738	-0.029906
Europe	0.064562	-0.209150	-0.112897	-0.295944	0.285738	1.000000	-0.151197
political.knowledge	-0.046598	-0.023510	-0.038528	-0.021299	-0.029906	-0.151197	1.000000

Table1.4 Correlation Matrix

Heatmap:

Multicollinearity is the important thing which can harm the model. Heatmap is a good way of identifying this issue.

Heatmap gives us the basic idea of relationship the variables have with each other.

Observations:

1.Highest positive correlation is between "economic.cond.national" and "economic.cond.household"

2. Highest negative correlation is between "Blair" and "Europe"

But, both of them are not huge.

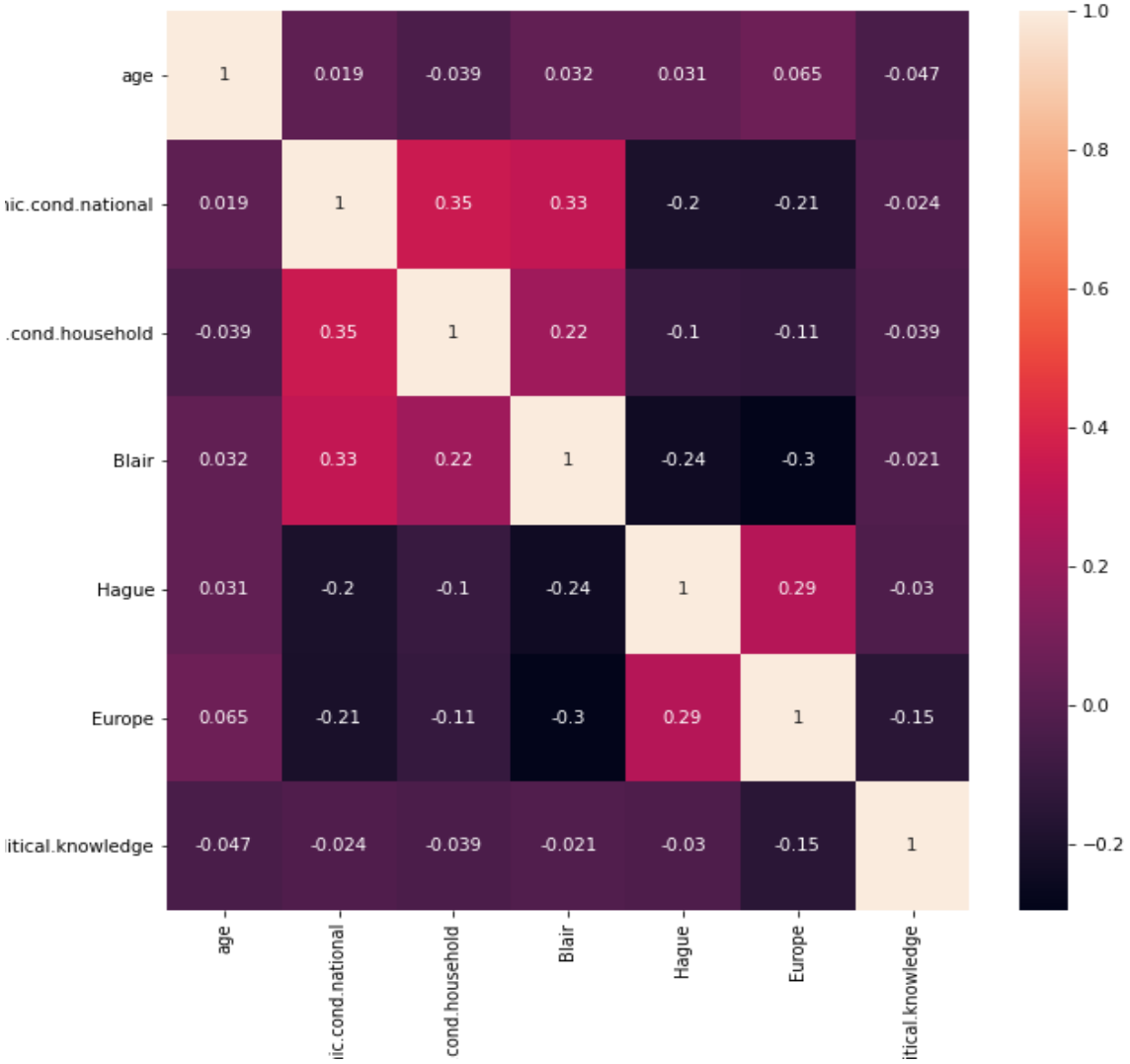


Fig8. Heatmap

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 DATASET:

As many Machine Learning models cannot work with string values, we will encode the categorical variables and convert their datatypes to integer datatype.

From the info of the dataset, we know that there are 2 categorical type variables, so we need to encode these 2 variables with the suitable technique.

Gender distribution:

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

Vote Distribution:

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

From the above results we can see that both variables contain only two classifications of data in them. We can use a simple categorical conversion (pd.categorical () or dummy encoding with drop_first = True, both of them will work here.) This will convert the values into 0 and 1. As there is no level or order in subcategory, any encoding will give the same result.

The datatype after conversion is int8 format, we can convert these to int64 format, it will work even if we don't change it to int64.

After Encoding:

Info-

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1517 entries, 0 to 1524
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   age                     1517 non-null   int64
 1   economic.cond.national  1517 non-null   int64
 2   economic.cond.household 1517 non-null   int64
 3   Blair                   1517 non-null   int64
 4   Hague                   1517 non-null   int64
 5   Europe                  1517 non-null   int64
 6   political.knowledge     1517 non-null   int64
 7   vote_Labour             1517 non-null   uint8
 8   gender_male             1517 non-null   uint8
dtypes: int64(7), uint8(2)
memory usage: 130.1 KB
```

Data:

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

Now, the model can build on this data.

Scaling is not necessary for this data. I am not scaling the data for this problem.

Datasplit: Splitting the data into test and train:

Before splitting the data, we need to find the target variable. Here, the target variable is "vote".

Vote data distribution:

```
Labour          0.69677
Conservative    0.30323
Name: vote, dtype: float64
```

There is a data imbalance in the variable as seen above. So, we cannot split it in 50:50 ratio. Instead, we can split the data into 70:30 ratio.

Now splitting both x and y data in the ratio 70:30, where train data is 70% and the test data is 30%.

After splitting- the shape of the data

The training set for independent variables: (1061,8)

The training set for dependent variables: (1061,)

The test set for the independent variable: (456,8)

The test set for the dependent variable: (456,)

Here,

X_train→ denotes 70% training dataset with 8 columns (except the target column called "vote").

X_test→ denotes 30% test dataset with 8 columns (except the target column "vote").

Y_train→ denotes the 70% training dataset with only the target column "vote"

Y_test→ denotes 30% test dataset with only target column "vote".

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

Logistic Regression Model:

Before fitting the model, it is important to know about the hyper parameters that is involved in model building.

Logistic Regression classifier: In the multiclass case, the training algorithm uses the one-vs-rest (OvR)scheme if the 'multi_class' option is set to 'ovr', and uses the cross-entropy loss if the 'multi_class' option is set to 'multinomial'. (Currently the 'multinomial' option is supported only by the 'lbfgs',

'sag', 'saga' and 'newton-cg' solvers.)

This class implements regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag', 'saga' and 'lbfgs' solvers. **Note that regularization is applied by default**. It can handle both dense and sparse input. Use C-ordered arrays or CSR matrices containing 64-bit floats for optimal performance; any other input format will be converted.

The 'newton-cg', 'sag', and 'lbfgs' solvers support only L2 regularization with primal formulation, or no regularization. The 'liblinear' solver supports both L1 and L2 regularization, with a dual formulation only for the L2 penalty. The Elastic-Net regularization is only supported by the 'saga' solver.

Parameters I have used:

solver= 'newton-cg'

max_iter = 10000

penalty = 'none'

verbose = True

→**penalty:** {'l1', 'l2', 'elasticnet', 'none'}, default='l2'

Specify the norm of the penalty:

- `'none'`: no penalty is added;

→**solver:** {'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'}, default='lbfgs'

Algorithm to use in the optimization problem. Default is 'lbfgs'. To choose a solver, you might want to consider the following aspects:

- For small datasets, 'liblinear' is a good choice, whereas 'sag' and 'saga' are faster for large ones;

- For multiclass problems, only 'newton-cg', 'sag', 'saga' and 'lbfgs' handle multinomial loss;

- 'liblinear' is limited to one-versus-rest schemes.

.. warning:

The choice of the algorithm depends on the penalty chosen:

Supported penalties by solver:

- 'newton-cg' - ['l2', 'none']

- 'lbfgs' - ['l2', 'none']

- 'liblinear' - ['l1', 'l2']

- 'sag' - ['l2', 'none']

- 'saga' - ['elasticnet', 'l1', 'l2', 'none']

→**max_iter**: int, default=100

Maximum number of iterations taken for the solvers to converge.

Linear Discriminant Analysis:

A classifier with a linear decision boundary, generated by fitting class conditional densities to the data and using Bayes' rule.

The model fits a Gaussian density to each class, assuming that all classes share the same covariance matrix.

The fitted model can also be used to reduce the dimensionality of the input by projecting it to the most discriminative directions, using the `transform` method.

Parameters which can be used:

LinearDiscriminantAnalysis(

solver='svd',

shrinkage=None,

priors=None,

n_components=None,

store_covariance=False,

tol=0.0001,

covariance_estimator=None,

)

To find the best combination among these parameters we will use the “GridSearchCV” method. This method can perform multiple combinations of these parameters simultaneously and can provide us with the best optimum results.

After Applying both Logistic Regression & LDA, I have loaded the train & test, accuracy, precession, recall & f1 scores into a data frame given below:

	precision train	precision test	recall train	recall test	f1 train	f1 test	accuracy train	accuracy test	train score	test score
Logistic Regression	0.860728	0.870130	0.909814	0.884488	0.884591	0.877250	0.831291	0.835526	0.831291	0.835526
Linear Discriminant Analysis	0.864899	0.864952	0.908488	0.887789	0.886158	0.876221	0.834119	0.833333	0.834119	0.833333

Inferences:

→Both the models performed really good.

→Compared to logistic Regression, the LDA model performed really very vell.

→There is no under-fitting or overfitting present, as the accuracy for both the test and train data are not very different. They look almost similar.

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

K Nearest Neighbours Model:

KNN is a distance based supervised machine learning algorithm that can be used to solve both classification and regression problems. Main disadvantage of this model is it becomes very slow when large volume of data is there and thus makes it an impractical choice when inferences need to be drawn quickly.

Before fitting the model, it is important to know about the hyperparameters that is involved in model building.

Parameters:

n_neighbours

weights

algorithm

To find the best combination among these parameters we will use the "GridSearchCV" method. This method can perform multiple combinations of these parameters simultaneously and can provide us with the best optimum results.

Naive Bayes Model:

Naïve Bayes classifiers is a model based on applying Bayes' theorem with strong(naïve) independent assumptions between the features. These assumptions however may not be the perfect case in real life scenarios.

Bayes Theorem-

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

Here the method that we are going to use is the GaussianNB() method, also known as BernoulliNB(). This method requires all the features to be in categorical type. A general assumption in this method is the data is following a normal or Gaussian distribution.

There are no specific parameters in this model like other, so we will simply fit the model with default parameters.

I have applied both the KNN & Naïve bayes without parameters only, after performing Gridserch & tuning, I have done models with the tuned parameters.

After performing both the models, I have loaded the train & test, accuracy, precession, recall & f1 scores into a data frame given below:

	precision train	precision test	recall train	recall test	f1 train	f1 test	accuracy train	accuracy test	train score	test score
KNN Model	0.877039	0.840979	0.927056	0.907591	0.901354	0.873016	0.855796	0.824561	0.855796	0.824561
Naive Bayes Model	0.875486	0.865132	0.895225	0.867987	0.885246	0.866557	0.835061	0.822368	0.835061	0.822368

Inference:

For KNN, the overall model performed well but there can be slight overfitting as accuracy is more for Train set than for the test.

For Naïve bayes, the model performed well on the data. There is no under-fitting or overfitting present, as the accuracy for both the test and train data are not very different. They look almost similar.

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

Model Tuning:

Tuning is a process of maximizing a model's performance without overfitting or creating too high of a variance. In machine learning, this is accomplished by selecting appropriate "hyper-parameters".

Grid Search is one of the most common methods of optimising the parameters. In this a set of parameters is defined and then the performance for each combination of these parameters is evaluated, using cross validation.

Then from among these models such as Bagging, Boosting Gradient boosting etc are prone to over fitting of data. Overfitting means that the model works very well on train data but works relatively poor in the test data. Under-fitting means that the model works very well on the test data, but works relatively poor on the training data.

Bagging Model (Using Random Forest Classifier):

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 the most commonly used with tree-based algorithms. It is a parallel method.

Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N data from the training. Training set for each of the base classifiers is independent of each other.

Hyper parameters which can be used in the model are:

max_depth

max_features

min_samples_leaf

min_samples_split

n_estimators

we can perform Grid Search with any of these parameters to get the best parameters.

Boosting Model:

Boosting is also an ensemble technique. It converts weal learners to strong learners. Unlike bagging it is a 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 a single strong prediction rule.

Misclassified input data gain a higher weight and examples that are classified correctly will lose weight. Thus, future weak learners focus more on the examples that previous weak learners misclassified They are also tree-based methods. There are many kinds of Boosting Techniques available and for this project, the following boosting techniques are to be used.
1. ADA Boost (Adaptive Boosting)

2. Gradient Boosting

3. Extreme Gradient Boosting

4. CAT Boost (Categorical Boosting)

Among these I am using the AdaBoosting Model.

ADA Boost (Adaptive Boosting):

This model is used to increase the efficiency of binary classifiers, but now used to improve multiclass classifiers as well. AdaBoost can be applied on top of any classifier method to learn from its issues and bring about a more accurate model and thus it is called the "best out-of-the -box classifier.

Before fitting the model, it is important to know about the hyper-parameters that is involved in model building.

Parameters:

algorithm

n_estimators

There are other parameters as well but we will use these for gridsearch, rest default values. Now after performing the "GridSearch CV, the best parameters obtained are

Algorithm = ‘SAMME’

n_estimators = 300

After performing both the models, I have loaded the train & test, accuracy, precession, recall & f1 scores into a data frame given below:

	precision train	precision test	recall train	recall test	f1 train	f1 test	accuracy train	accuracy test	train score	test score
Bagging Model	0.961538	0.848297	0.994695	0.904290	0.977836	0.875399	0.967955	0.828947	0.967955	0.828947
Ada Boost Model	0.880922	0.842767	0.912467	0.884488	0.896417	0.863124	0.850141	0.813596	0.850141	0.813596

Inference:

In Bagging Model, we can see that there is an overfitting as accuracy is more for Train set than for the test set. There is a huge difference in accuracy scores for train and test.

Hence, bagging is overfitting model.

For Boosting, the overall model performed well. But there can be slight overfitting as the accuracy is more for train set than for test set.

The Best parameters we got after Model Tuning are:

The parameters I have used for applying GridSearch on **Logistic regression** are:

```
GridSearchCV(cv=3, estimator=LogisticRegression(max_iter=50000, random_state=1),
             param_grid={'penalty': ['l2', 'none'],
                         'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag',
                                    'saga'],
                         'tol': [0.0001, 1e-05]})
```

The best parameters are:

{'penalty': 'l2', 'solver': 'liblinear', 'tol': 0.0001}

LDA:

```
GridSearchCV(cv=3, estimator=LinearDiscriminantAnalysis(),
             param_grid={'solver': ['svd', 'lsqr', 'eigen'],
                         'tol': [0.001, 0.0001, 1e-05]})
```

The best parameters are:

{'solver': 'svd', 'tol': 0.001}

KNN:

```
GridSearchCV(cv=3, estimator=KNeighborsClassifier(),
             param_grid={'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
                         'leaf_size': [15, 20, 25, 30, 35],
                         'n_neighbors': [2, 3, 7, 9, 11],
                         'weights': ['uniform', 'distance']})
```

Best parameters are:

{'algorithm': 'auto', 'leaf_size': 25, 'n_neighbors': 7, 'weights': 'uniform'}

Bagging:

```
GridSearchCV(cv=3,
             estimator=BaggingClassifier(base_estimator=RandomForestClassifier(random_state=1),
                                         random_state=1, verbose=False),
             param_grid={'n_estimators': [300, 500]})
```

Best parameters are:

{'n_estimators': 300}

AdaBoost:

```
GridSearchCV(cv=3, estimator=AdaBoostClassifier(random_state=1),
             param_grid={'algorithm': ['SAMME', 'SAMME.R'],
                         'n_estimators': [200, 300, 500]})
```

Best parameters are:

{'algorithm': 'SAMME', 'n_estimators': 300}

I have performed all the models with the best parameters after tuning the models.

1.7Performance 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.

Performance Metrics:

Usually there are many performance metrics that are used in assessing the strength of the model to understand how the model has performed as well as to take an informed decision on whether to go forward with the model in the real time scenario or not.

The Industrial standards are generally based on the following methods:

→Classification Accuracy

→Confusion Matrix

→Classification Report

→Area Under ROC Curve (Visualisation) and AUC score.

Logistic Regression:

Train Accuracy: 0.836946

Test Accuracy: 0.828947

Confusion Matrix:

For train data

True negative: 198

False Negative: 64

False Positive: 109

True Negative: 690

For test data:

True negative: 110

False Negative: 35

False Positive: 43

True Negative: 268

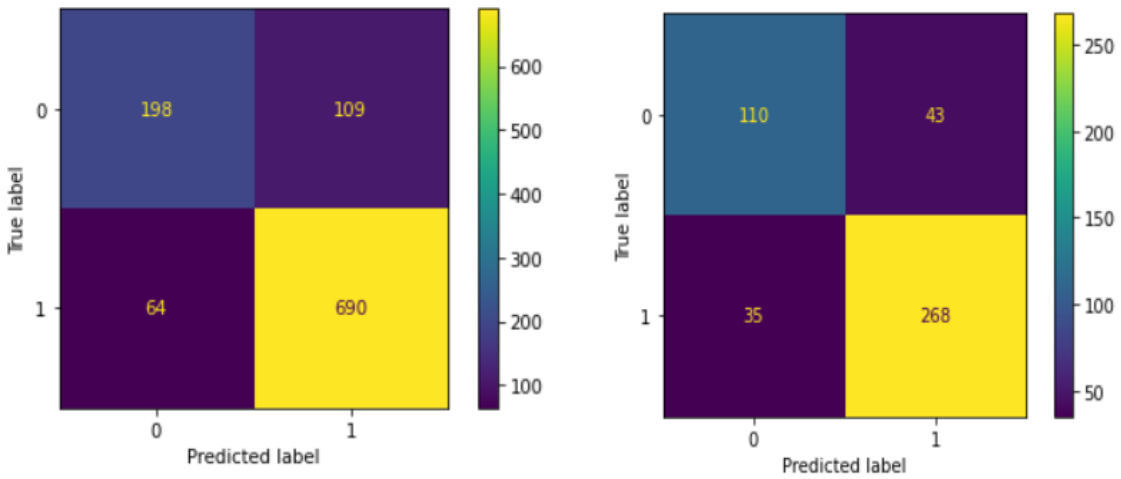


Fig9. Confusion matrix -LR

Classification Report:

Train:

```
TRAIN DATA
Model Name:  LogisticRegression(max_iter=50000, solver='liblinear')
              precision    recall  f1-score   support

           0       0.76      0.64      0.70       307
           1       0.86      0.92      0.89       754

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

Test:

```
Test DATA
Model Name:  LogisticRegression(max_iter=50000, solver='liblinear')
              precision    recall  f1-score   support

           0       0.76      0.72      0.74       153
           1       0.86      0.88      0.87       303

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

Area Under ROC Curve and AUC Score:

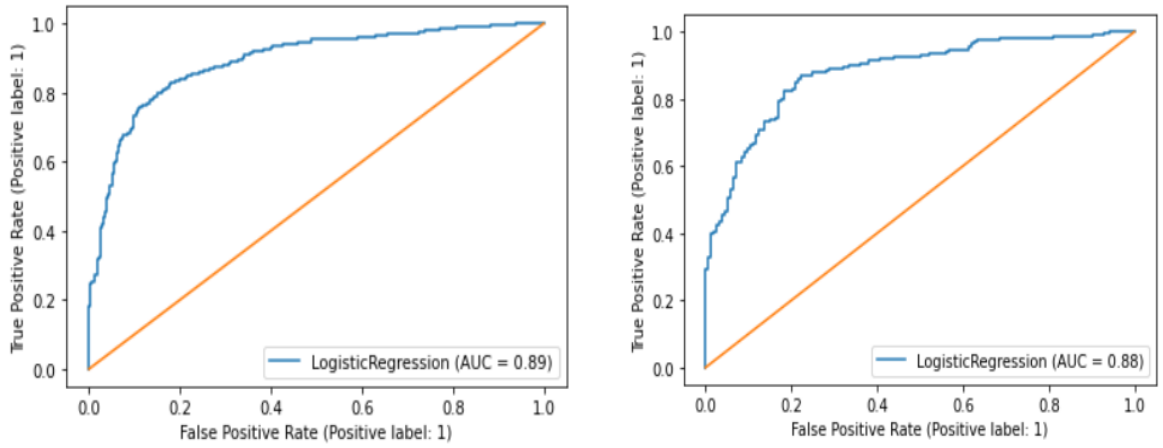


Fig10. AUC_ROC Train test - LR

LDA (Linear Discriminant Analysis):

Train Accuracy: 0.834119

Test Accuracy: 0.833333

Confusion Matrix:

For train data

True negative: 200

False Negative: 69

False Positive: 107

True Negative: 685

For test data:

True negative: 111

False Negative:34

False Positive: 42

True Negative:269

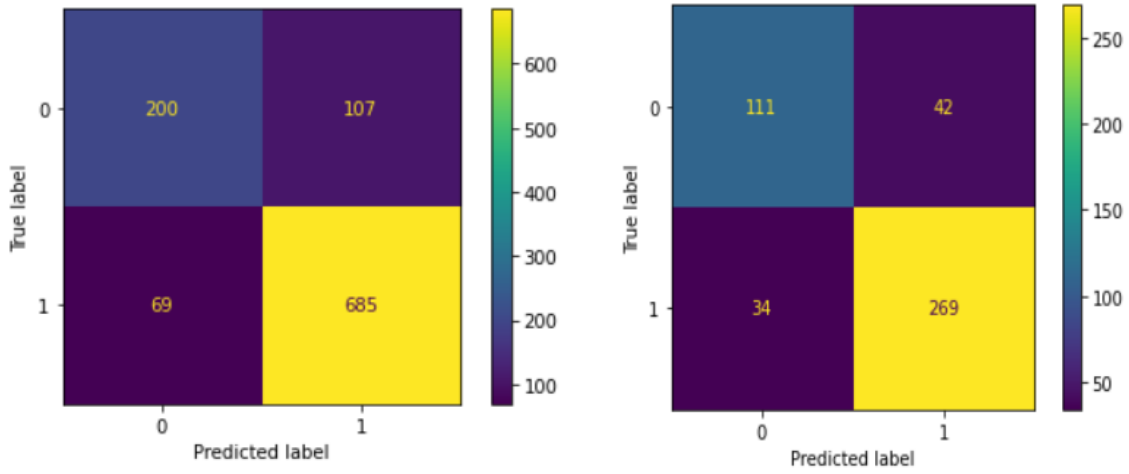


Fig11. Confusion Matrix—LDA

The model performs well if the False positive and False negative values are very small and less than True positives and True negatives.

Classification Report:

Train:

```
TRAIN DATA
Model Name:  LinearDiscriminantAnalysis(tol=0.001)
              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
```

Test:

```
Test DATA
Model Name:  LinearDiscriminantAnalysis(tol=0.001)
              precision    recall  f1-score   support

           0       0.77      0.73      0.74       153
           1       0.86      0.89      0.88       303

    accuracy                           0.83       456
   macro avg       0.82      0.81      0.81       456
weighted avg       0.83      0.83      0.83       456
```

Area Under ROC Curve and AUC Score:

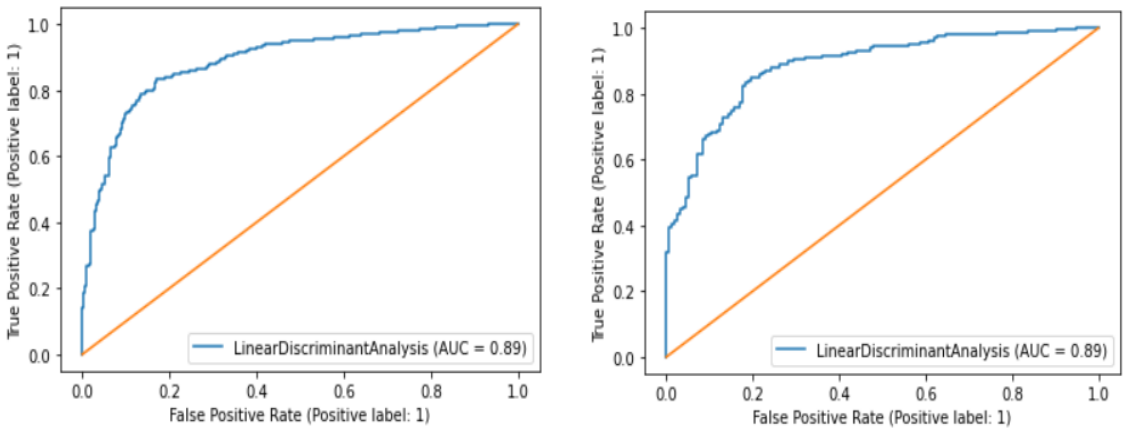


Fig12. AUC_ROC Train test - LDA

KNN:

Train Accuracy: 1.000000

Test Accuracy: 0.817982

Confusion Matrix:

For train data

True negative: 307

False Negative:0

False Positive: 0

True Negative:754

For test data:

True negative:99

False Negative:29

False Positive: 54

True Negative:274

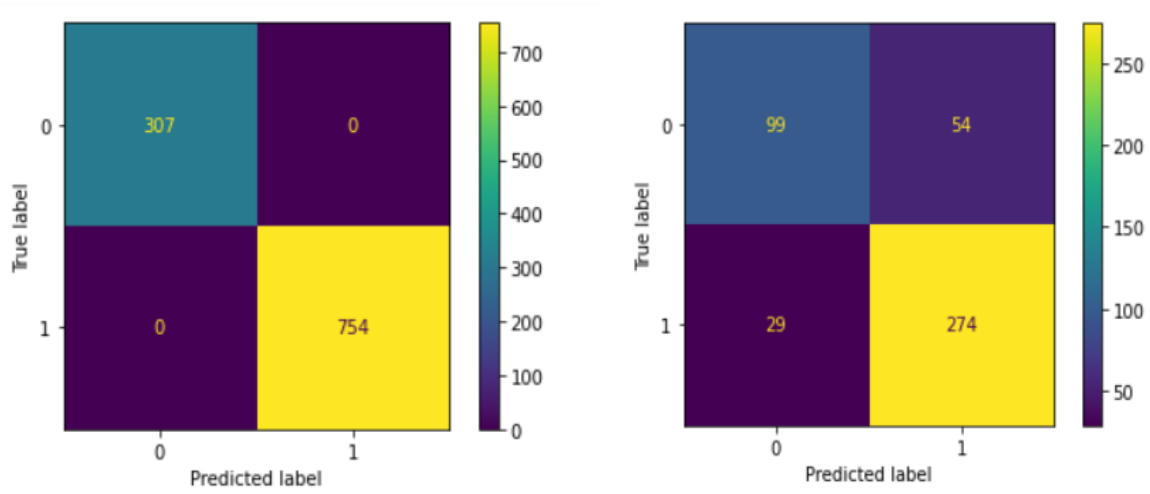


Fig13. Confusion Matrix—KNN

Classification Report:

```
TRAIN DATA
Model Name:  KNeighborsClassifier(algorithm='brute', leaf_size=15, n_neighbors=9,
                  weights='distance')
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       307
           1       1.00      1.00      1.00       754

    accuracy                           1.00      1061
   macro avg       1.00      1.00      1.00      1061
weighted avg       1.00      1.00      1.00      1061
```

```
Test DATA
Model Name:  KNeighborsClassifier(algorithm='brute', leaf_size=15, n_neighbors=9,
                  weights='distance')
              precision    recall  f1-score   support

           0       0.77      0.65      0.70       153
           1       0.84      0.90      0.87       303

    accuracy                           0.82       456
   macro avg       0.80      0.78      0.79       456
weighted avg       0.81      0.82      0.81       456
```

Area Under ROC Curve and AUC Score:

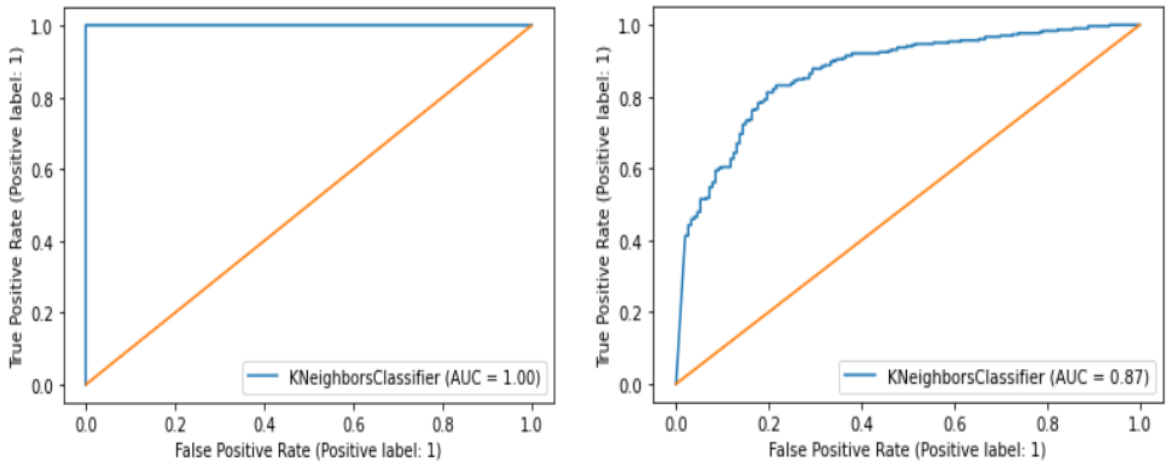


Fig14. AUC_ROC Train test - KNN

Naïve Bayes Model:

Train Accuracy: 0.835061

Test Accuracy: 0.822368

Confusion Matrix:

For train data

True negative: 211

False Negative:79

False Positive: 96

True Negative:675

For test data:

True negative:112

False Negative:40

False Positive: 41

True Negative:263

Confusion Matrix:

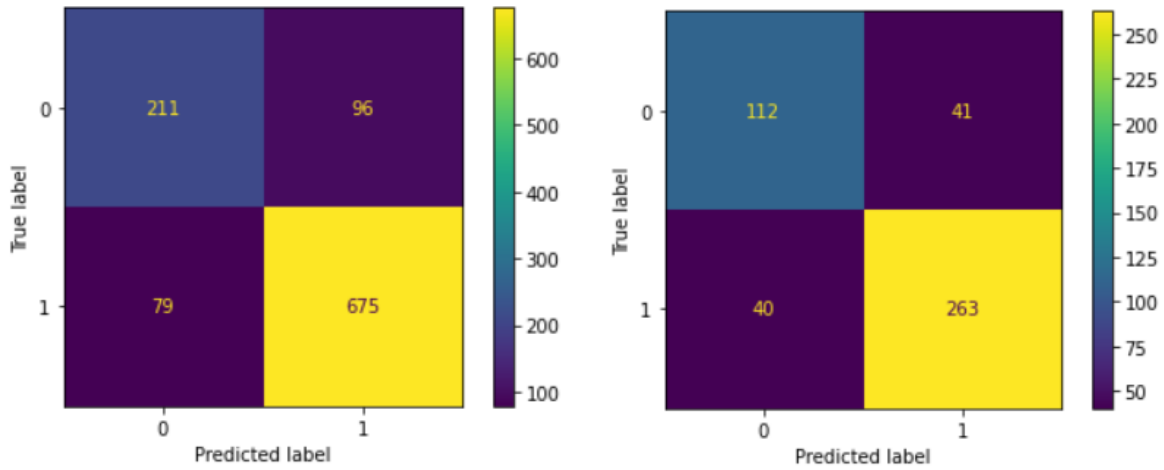


Fig15. Confusion Matrix—Naïve Bayes

Classification Report:

Train:

TRAIN DATA
Model Name: GaussianNB()

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

Test:

TEST DATA
Model Name: GaussianNB()

	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

Area Under ROC Curve and AUC Score:

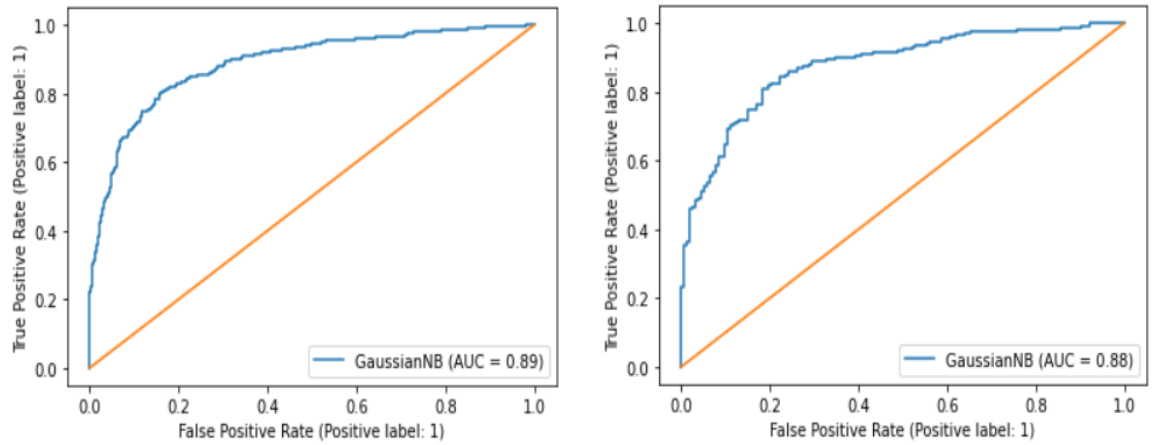


Fig16. AUC_ROC Train test – Naïve Bayes

Bagging:

Train Accuracy: 0.967012

Test Accuracy: 0.833333

Confusion Matrix:

For train data

True negative: 276

False Negative:4

False Positive: 31

True Negative:750

For test data:

True negative:104

False Negative:49

False Positive: 27

True Negative:276

Confusion Matrix:

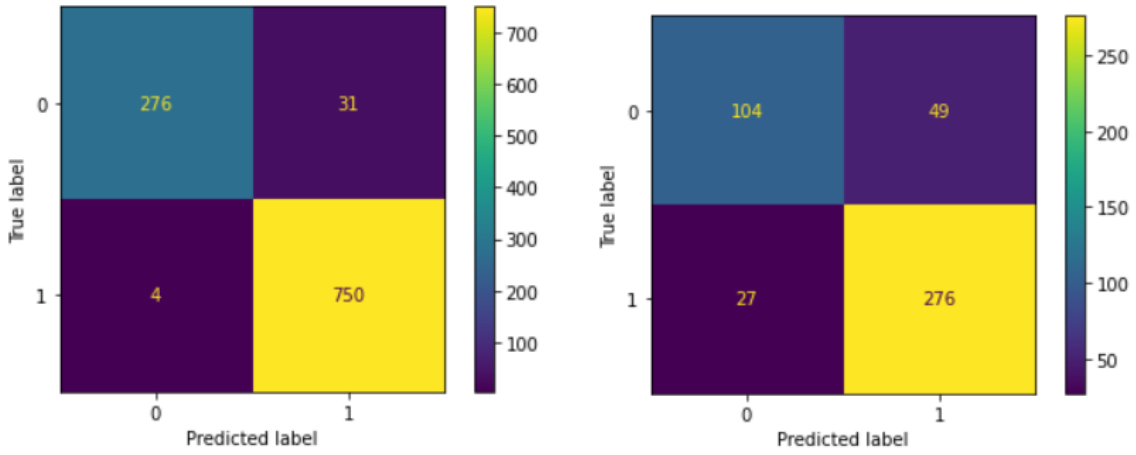


Fig17. Confusion Matrix—Bagging

Classification Report:

Train:

```
TRAIN DATA
Model Name:  BaggingClassifier(base_estimator=RandomForestClassifier(random_state=1),
                  n_estimators=300, random_state=1)
              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
```

Test:

```
Test DATA
Model Name:  BaggingClassifier(base_estimator=RandomForestClassifier(random_state=1),
                  n_estimators=300, random_state=1)
              precision    recall  f1-score   support

           0       0.79      0.68      0.73       153
           1       0.85      0.91      0.88       303

    accuracy                           0.83       456
   macro avg       0.82      0.80      0.81       456
weighted avg       0.83      0.83      0.83       456
```

Area Under ROC Curve and AUC Score:

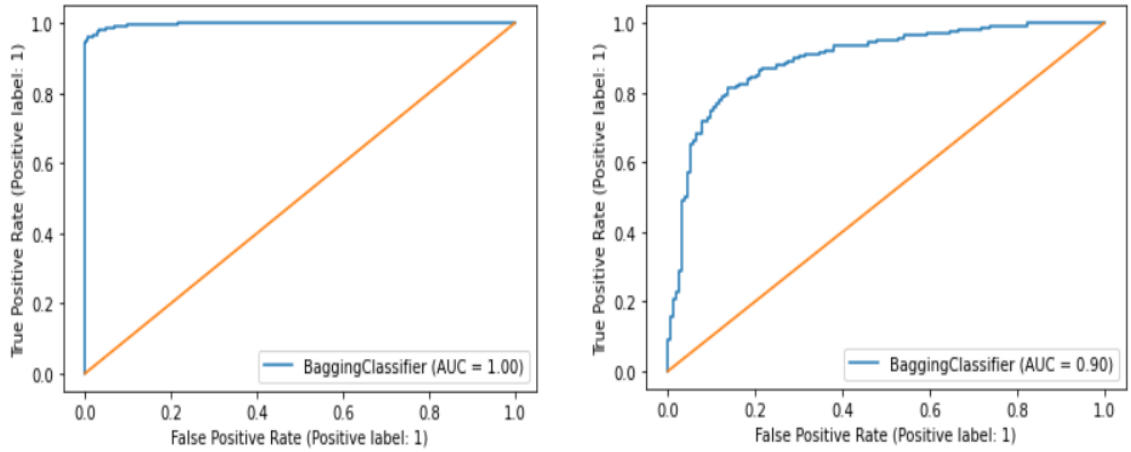


Fig18. AUC_ROC Train test – Bagging

AdaBoosting:

Train Accuracy: 0.845429

Test Accuracy: 0.822368

Confusion Matrix:

For train data

True negative: 206

False Negative:63

False Positive: 101

True Negative:691

For test data:

True negative:108

False Negative:36

False Positive: 45

True Negative:267

Confusion Matrix:

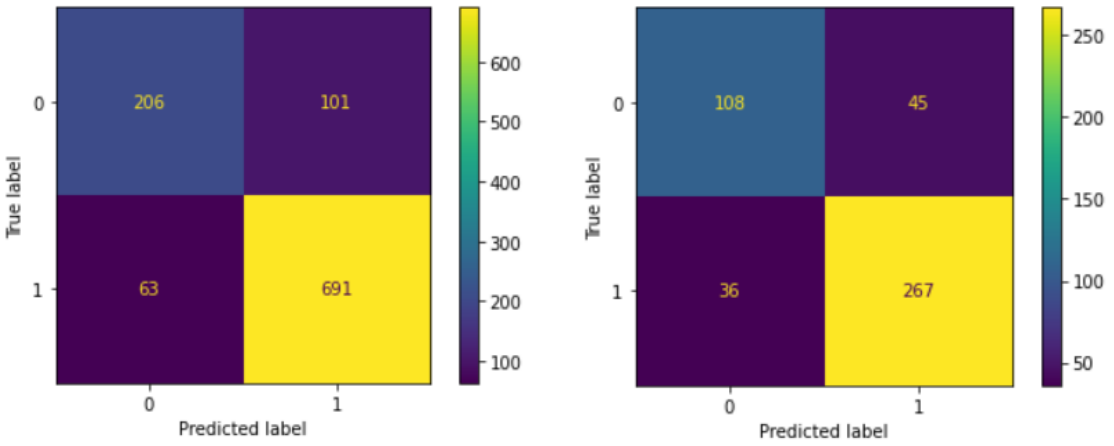


Fig19. Confusion Matrix—Adaboost

Classification Report:

Train:

TRAIN DATA
Model Name: AdaBoostClassifier(algorithm='SAMME', n_estimators=300, random_state=1)

	precision	recall	f1-score	support
0	0.77	0.67	0.72	307
1	0.87	0.92	0.89	754
accuracy			0.85	1061
macro avg	0.82	0.79	0.80	1061
weighted avg	0.84	0.85	0.84	1061

Test:

Test DATA
Model Name: AdaBoostClassifier(algorithm='SAMME', n_estimators=300, random_state=1)

	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	0.80	0.79	0.80	456
weighted avg	0.82	0.82	0.82	456

Area Under ROC Curve and AUC Score:

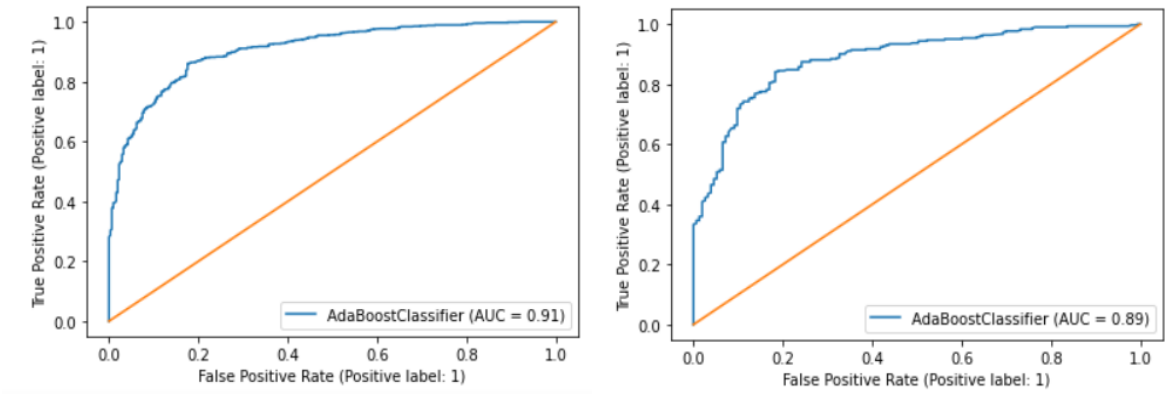


Fig20. AUC_ROC Train test – AdaBoost

Model Comparison:

This is a process through which we will compare all models build and find the best optimised among all the models. Here, I am comparing the six models I have trained before tuning and after tuning the model.

The basis on which models are evaluated are known as performance metrics.

The metrics on which model will be evaluated are:

→Accuracy

→AUC

→Precision

→F1-Score

I have made a data frame with all the accuracy, Precession, Recall and F1 s-scores of all the 6 models before tuning and after tuning.

	precision train	precision test	recall train	recall test	f1 train	f1 test	accuracy train	accuracy test	train score	test score
Logistic Regression	0.860728	0.870130	0.909814	0.884488	0.884591	0.877250	0.831291	0.835526	0.831291	0.835526
Linear Discriminant Analysis	0.864899	0.864952	0.908488	0.887789	0.886158	0.876221	0.834119	0.833333	0.834119	0.833333
KNN Model	0.877039	0.840979	0.927056	0.907591	0.901354	0.873016	0.855796	0.824561	0.855796	0.824561
Naive Bayes Model	0.875486	0.865132	0.895225	0.867987	0.885246	0.866557	0.835061	0.822368	0.835061	0.822368
Bagging Model	0.961538	0.848297	0.994695	0.904290	0.977836	0.875399	0.967955	0.828947	0.967955	0.828947
Ada Boost Model	0.880922	0.842767	0.912467	0.884488	0.896417	0.863124	0.850141	0.813596	0.850141	0.813596
Lgr_gs	0.863579	0.861736	0.915119	0.884488	0.888603	0.872964	0.836946	0.828947	0.836946	0.828947
Lda_gs	0.864899	0.864952	0.908488	0.887789	0.886158	0.876221	0.834119	0.833333	0.834119	0.833333
Knn_gs	1.000000	0.835366	1.000000	0.904290	1.000000	0.868463	1.000000	0.817982	1.000000	0.817982
bagging_gs	0.960307	0.849231	0.994695	0.910891	0.977199	0.878981	0.967012	0.833333	0.967012	0.833333
Ada_Boost_gs	0.872475	0.855769	0.916446	0.881188	0.893920	0.868293	0.845429	0.822368	0.845429	0.822368

Table 1.5 Final Dataframe

From the above table:

Basis on **Accuracy** – Logistic Regression and LDA model performed really well compared to other models.

Basis on **AUC Score** – Both Logistic Regression and LDA model performed really well compared to other models.

Basis on **Recall** – KNN, LDA, Logistic regression models performed slightly better than others

Basis on **F1-Score**: Logistic Regression and LDA model performed really well compared to other models.

All the models performed well with slight difference ranging from (1-5%).

Observations:

All the models performed well with slight difference ranging from (1-5%).

Performance can be slightly improved, if applied after scaling the data for the Distance based algorithms.

Best Optimised Model – On the basis of all the comparisons and performance metrics. Both the LDA and Logistic regression performed well. Compared to both, I consider **Logistic Regression** as the best optimised model.

1.8 Based on these predictions, what are the insights?

Our main Business Objective is – "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."

→We have used Logistic Regression Model for predicting the outcome, as it has the best optimised performance compared to rest of the models.

→Hyper- parameter tuning is an important aspect of model building.

→There are limitations to hyper- parameter tuning, as to process these combinations huge amount of processing power is required.

→But if tuning is done with many sets of parameters, then we might get even better results.

→Gathering more data, and having good understanding about each feature in the dataset will help in training the models and helpful in improving their predictive powers.

→Boosting models can also perform well even without tuning. If we perform hyper-parameter tuning, there is a chance of getting 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.

Observation:

The business issue essentially spun around fostering a model to anticipate which party a citizen would vote in favor of depending on the data about the citizens. The model will in this way be utilized to make an exit poll that will help in predicting the overall win and seats covered by a specific party. For this to achieve, the analyses assumed CNBE wish to focus more on accurately predicting the Labor's win and hence that has been the class of choice for prediction. The analysis and building of Machine Learning models based on a restricted dataset of 1525 citizens with specific details of the electors. This notwithstanding, regardless of limitations, has assisted us with finding not many key bits of knowledge and patterns alongside exhibiting the ideal model which could be used by CNBE to anticipate the previously mentioned.

Insights summary:

→Majority of the voters are between the ages 33 – 75 and there are no voters' data capture between the age 18 to 24;

→Majority of people think that household and national economic condition is satisfactory as most have ranked them in 3 or 4 out of 5;

→Conservatives consists of slightly higher proportion of aged voters (50 years and above); 50% of the voters are of age above 53 years and only the bottom 25% voters are aged less than 41 years;

→Labour leader Blair is more popular among people than Conservative leader Hague as Blair has received a rating of 4 on average, whereas Hague has received a mixed rating of 2 and 4; The general population does not seem to be very eurosceptic as cumulative frequency of non-eurosceptic people (who opted for 6 or less) seem to be higher than the cumulative frequency of eurosceptic people (who opted for 7 or higher);

There are more female voters than male voters. Conservative voters have better political knowledge of political parties' position on European integration than their Labour counterparts. Labour voters appears to have a pro-European integration opinion as opposed to Conservative voters. 43% Conservative voters have rated the national economic condition average with score of 3, further indicating, the overall assessment to be between poor and average. Most people find Blair to be a better leader and if the Conservative party wants to win then they have to focus in improving Hague's image among people, or go with a different candidate;

Business Recommendations:

CNBE must gather data of voters aged between 18 and 24 so as to make the predictions more accurate; It needs to be addressed that, the larger the number of voters, better the Machine Learning models can be optimized;

The dataset must also include additional assessment ratings about migration policy including refugee settlement, employment generation, income tax regime, etc;

Problem 2:

EXCECUTIVE SUMMARY:

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

President Franklin D. Roosevelt in 1941

President John F. Kennedy in 1961

President Richard Nixon in 1973

	text	fileid
0	On each national day of inauguration since 178...	1941-Roosevelt.txt
1	Vice President Johnson, Mr. Speaker, Mr. Chief...	1961-Kennedy.txt
2	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	1973-Nixon.txt

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

Characters:

```
Number of characters in the fileid 1941-Roosevelt.txt: 7571
Number of characters in the fileid 1961-Kennedy.txt: 7618
Number of characters in the fileid 1973-Nixon.txt: 9991
```

Words:

```
Number of words in the fileid 1941-Roosevelt.txt: 1536
Number of words in the fileid 1961-Kennedy.txt: 1546
Number of words in the fileid 1973-Nixon.txt: 2028
```

Sentences:

```
Number of sentences in the fileid 1941-Roosevelt.txt: 68
Number of sentences in the fileid 1961-Kennedy.txt: 52
Number of sentences in the fileid 1973-Nixon.txt: 69
```

2.2 Remove all the stop words from all three speeches.

To remove stopwords, there is a package called **"Stopwords"** in the nltk.corpus library.

Hence, in order to do so we need to import following libraries-

→from nltk.corpus import stopwords

→from nltk.stem.porter import PortStemmer

The stopwords library contains all the stopwords like ‘and’,’a’,’is’,’.‘,’of’,’to’ etc., that usually don’t have any importance in understanding the sentiment or usefulness in Machine learning algorithms.

These stopwords present in the package are universally accepted stopwords and we can add using the (.extend()) function or can 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. Here, we will use English.

Stemming is a process which helps the processor in understanding the words that have similar meaning. In this, the words are brought down to their base or root level by removing the affixes. It is highly used in search engines.

For e.g. -eating, eats, eaten all these will be reduced to eat after stemming.

Some of the stopwords removed are:

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

'won', "won't", 'wouldn',"wouldn't",'!','"','#','$','%','&', "'",'(',')', '*','+',',','-','.','/', ':',';','<','=','>', '?','@','[','\\',d']','^','_','`','{','|','}','~','--','us','let']

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

Results after removing stopwords:

→**For Franklin D. Roosevelt's Speech:**

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

```
[('nation', 12),
 ('know', 10),
 ('spirit', 9),
 ('life', 9),
 ('democracy', 9),
 ('people', 7),
 ('america', 7),
 ('years', 6),
 ('freedom', 6),
 ('human', 5),
 ('men', 5),
 ('new', 5),
 ('body', 5),
 ('mind', 5),
 ('speaks', 5)]
```

Here 'spirit', 'life', 'democracy' are on 3rd place because of the same number of occurances.

Most occurring word: Nation

→**For John F. Kennedy's speech:**

```
[('world', 8), ('sides', 8), ('new', 7)]
```

Most occurring word: world

→**For Richard Nixon's speech:**

```
[('america', 21), ('peace', 19), ('world', 18)]
```

Most occurring word: america

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

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

→For generating word-cloud we need word-cloud package. By default, it is not installed in the kernel, so we have to install it.

→After importing the package, we will again remove the stopwords but will not perform stemming.

→As removing the stopwords would remove and filter the unwanted words that possibly have no sentiment analysis.

Word Cloud for Franklin D. Roosevelt's Speech:

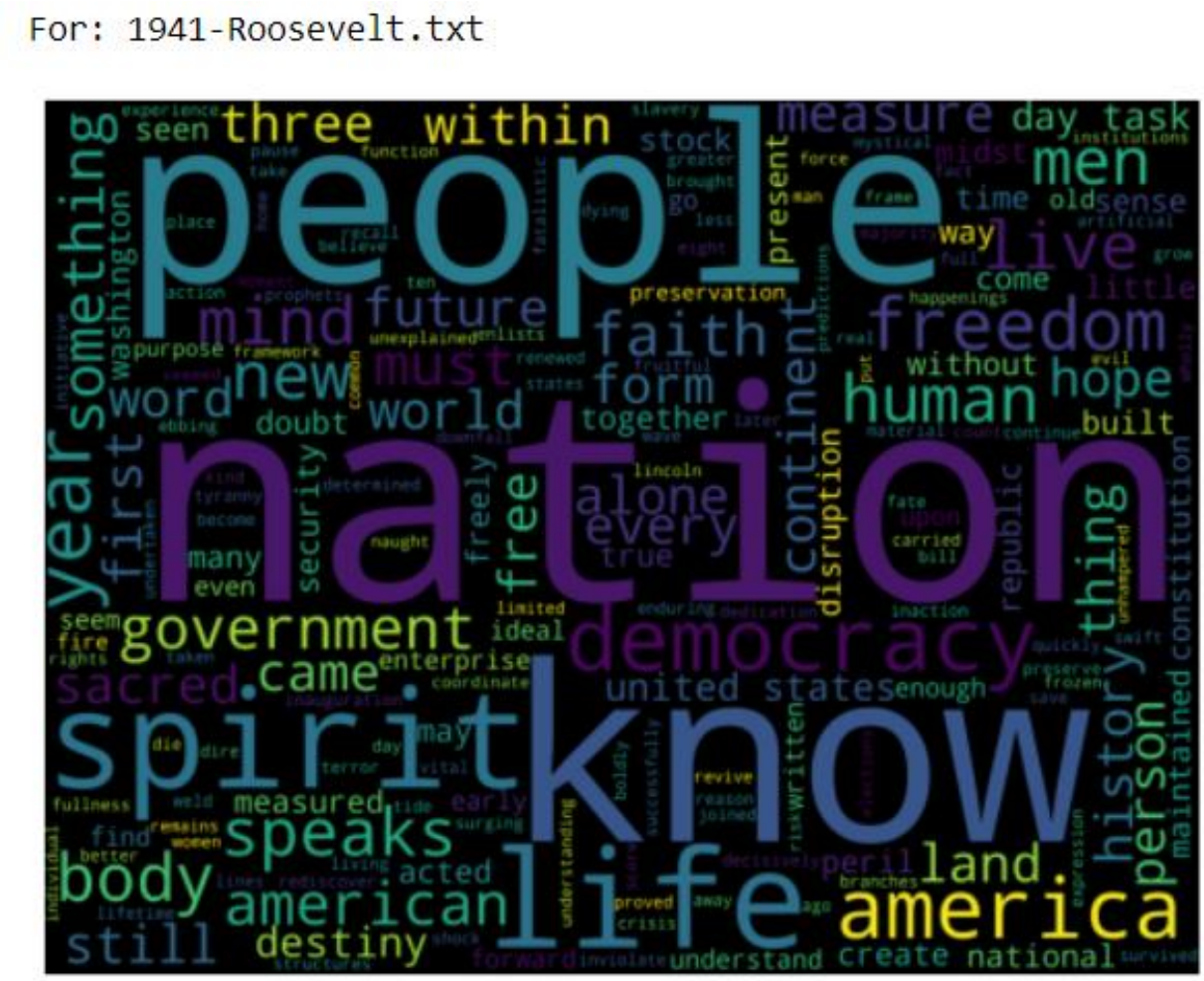


Fig21. Word Cloud for Franklin D. Roosevelt's Speech

We can see some highlighted words like "nation", "know", "people", etc. which we observed as top words in the previous question.

This shows the bigger size, more the frequency.

Word Cloud for John F. Kennedy's speech:

For: 1961-Kennedy.txt

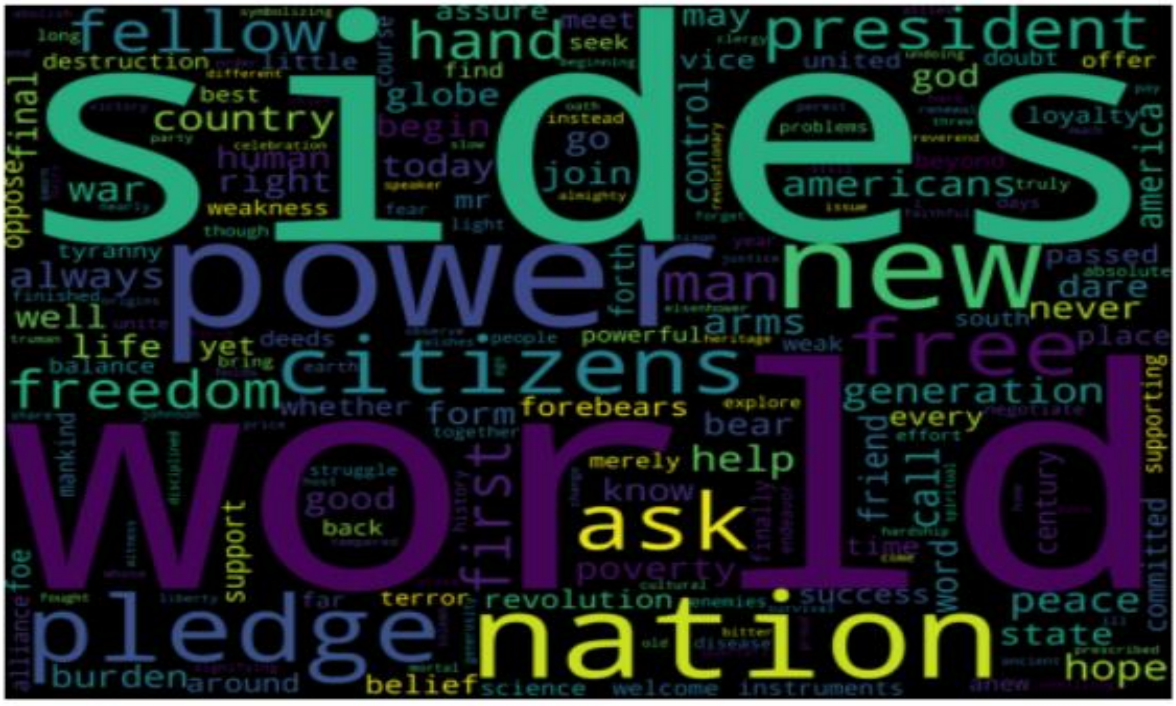


Fig22.Word Cloud for John F. Kennedy's speech

Word Cloud for Richard Nixon's speech:

For: 1973-Nixon.txt

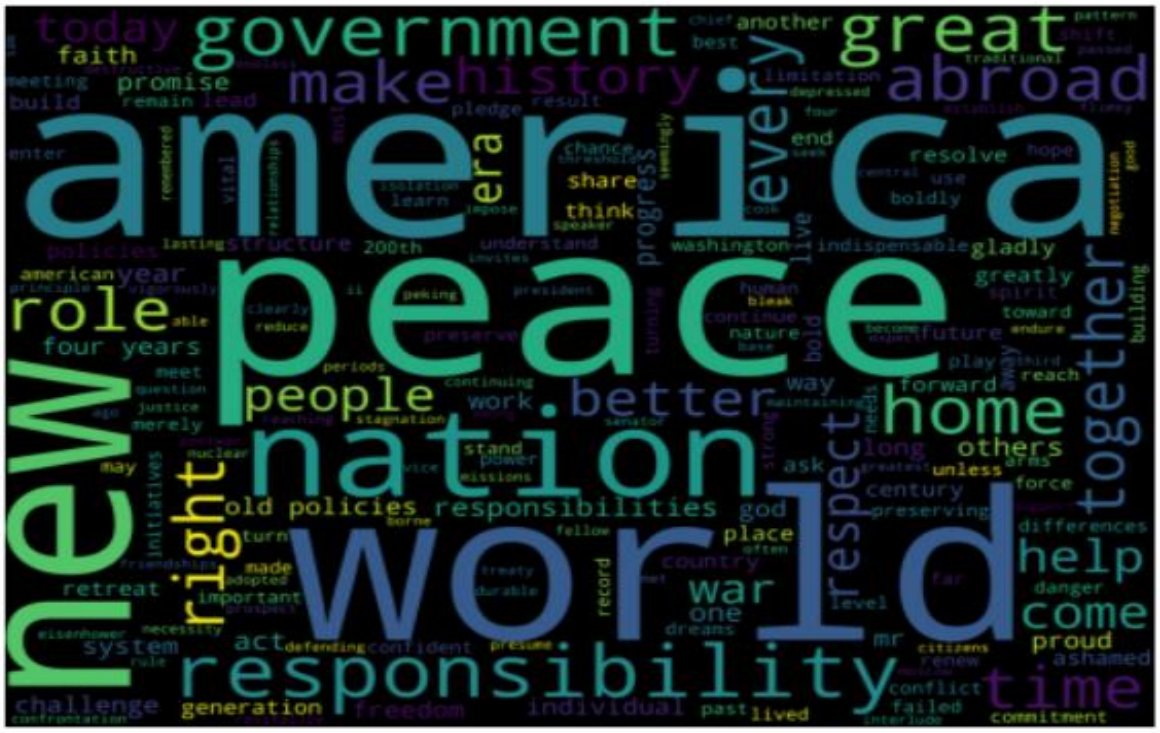


Fig23.Word Cloud for Richard Nixon's speech

Roosevelt had used the word Roosevelt had used the word ‘nation’ the most in hi ‘nation’ the most in his speeches followed by words s speeches followed by words such as ‘spirit’, ‘people’, ‘life’, ‘America’. Further, his speech a Further, his speech also stressed on positive words lso stressed on positive words like ‘spirit’, ‘secu’ like ‘spirit’, ‘security’, ‘life’, ‘faith’, rity’, ‘life’, ‘faith’, etc. Other prominent words visible are ‘live, ‘freedom’, ‘people’, ‘America’, ‘preserve’, ‘history’. Based on the word cloud, based on the word cloud, the sentiment of his speech the sentiment of his speech is positive; h is positive. His speech seems to encourage the His speech seems to encourage the audience to prese audience to preserve America’s history, democracy rve America’s history, democracy and freedom. The president is talking about the cou, The president is talking about the country which is ntry which is stressed through words like ‘nation’ stressed through words like ‘nation’ and ‘America’ and ‘people’.

Kennedy in his speech hav Kennedy in his speech have stressed on the words li e stressed on the words like ‘let’, ‘new’, ‘world’, ke ‘let’, ‘new’, ‘world’, ‘power’, ‘power’, ‘nation’, ‘side’, etc. Unlike Roosevelt's speeches, his speeches seem to center on 'power', 'nation', and 'world'. These words refer to a more agg These words refer to a more aggressive approach to ressive approach to build America as a new world po build America as a new world power. Also, unlike Roosevelt's speeches, the use of words such as "peace", "hope", "us", "human", "friend" and "citizen" are less common in his speeches. The word cloud implies that his speeches are for his audience to embrace and support the idea of America's global power, not idea of America's global power, not the actual well the actual well-being of its citizens. -being of its citizens. However, the above sentimental inference could however be challenged in different context.

Nixon in his speech is centered around the words like ‘let’, ‘us’, ‘peace’, ‘world’, ‘America’, ‘nation’, ‘role’, ‘government’, etc. Unlike Kennedy's speeches, the general idea of his speeches appears to be how people in the United States can contribute to world peace. His speeches, similar to Roosevelt's, give a positive vibration, which ve vibration, which is shown in repeated is shown in repeated uses of words such as "live," "build," uses of words such as "live," "build," "right," "to "right," "together," "promise," and "justice”. gether," "promise," and "justice”; The word cloud thus suggests the ov The word cloud thus suggests the overall positive e erall positive encouragement for the audience; ncouragement for the audience. It is interesting to note that Nixon's speech takes a more balanced approach a more balanced approach to the nation- to the nation- building and their positive impacts in a global context compared to Roosevelt's speech, where the whole message focused o where the whole message focused on the human aspect n the human aspects of the country. It is derived f s of the country. It is derived from common words such as 'peace', common words such as 'peace', ‘home’, 'nation', 'to ‘home’, 'nation', 'together', 'faith', 'justice'. gether', 'faith', 'justice'.

Insights:

→Our Objective was to look at all the three speeches and analyse them. To find the strength and sentiment of the speeches.

→Based on the outputs we can see that there are some similar words that are present in all speeches.

→these words may be the point which inspired many people and also get them the seat of the president of United States of America.

→Among all these speeches **“nation”** is the word that is significantly highlighted in all the three speeches.