PG-DSBA Dec_A 2020

Machine Learning Project

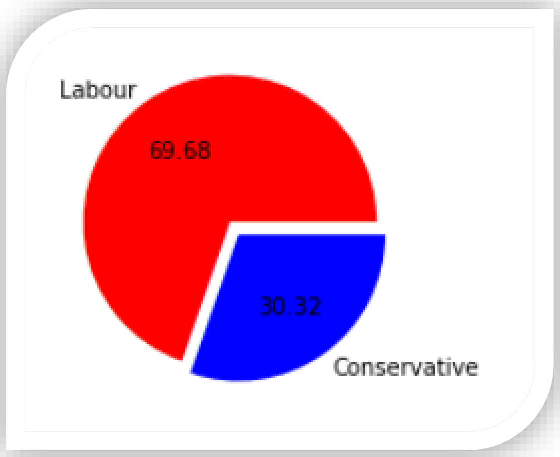
Submitted by: F Maria Jasmine

Date of Submission: 11/07/2021

Table of Contents

II. Problem Statement: Text Analytics – Inaugural Speeches

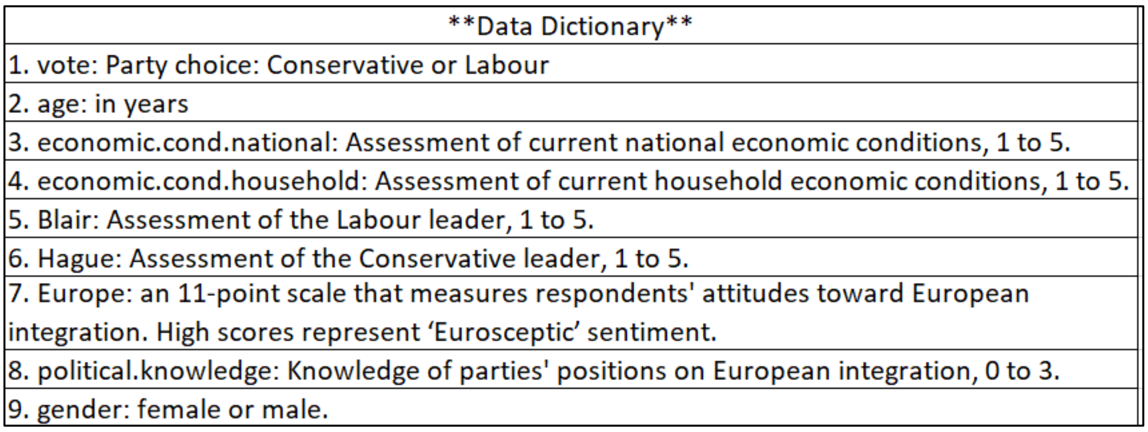
Labour or Conservative



***Problem 1**: Election Vote Prediction using Binary classifiers:*

You are hired by one of the leading news channels CNBE who wants to analyse recent elections. This survey was conducted on 1525 voters with 9 variables. You have to build a model, to predict which party a voter will vote for on the basis of the given information, to create an exit poll that will help in predicting overall win and seats covered by a particular party.

Data Dictionary:



Data Dictionary
1. vote: Party choice: Conservative or Labour
2. age: in years
3. economic.cond.national: Assessment of current national economic conditions, 1 to 5.
4. economic.cond.household: Assessment of current household economic conditions, 1 to 5.
5. Blair: Assessment of the Labour leader, 1 to 5.
6. Hague: Assessment of the Conservative leader, 1 to 5.
7. Europe: an 11-point scale that measures respondents' attitudes toward European integration. High scores represent 'Eurosceptic' sentiment.
8. political.knowledge: Knowledge of parties' positions on European integration, 0 to 3.
9. gender: female or male.

Problem statement:

Using the Binary classifier models, 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.

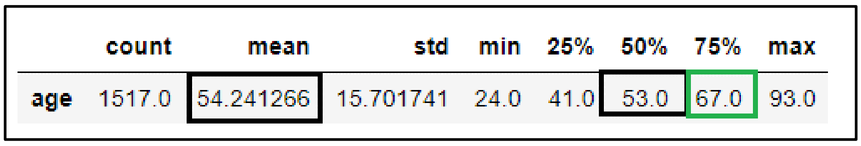
1.1 Read the dataset. Do the descriptive statistics and do the null value condition check. Write an inference on it. (4 Marks)

Data Ingestion:

The Election data set has 1525 rows and 9 features excluding the Un-named column.

1. There are 8 duplicate data in the data. Duplicates are dropped.
2. No null values in the given data set
3. The given data has 1 continuous variable and 6 integer categorical variable and 2 object type categorical variable.
4. No anomalies in the given dataset.
5. Numerical Data summary:

Table: 1.1: Numerical Description Summary

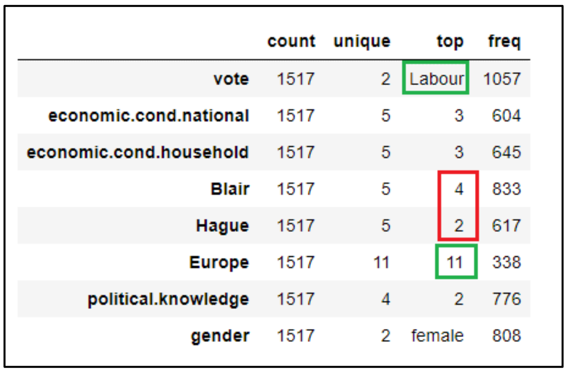


	count	mean	std	min	25%	50%	75%	max
age	1517.0	54.241266	15.701741	24.0	41.0	53.0	67.0	93.0

- The mean age and the median age are almost same. No possible outliers in the age feature.
- 75% of the voters are aged below 70. Only 4% of the voters are aged above 80. As there are no outliers, no changes are made to age feature.

6. Discrete Data summary.

Table 1.2: Discrete Description summary



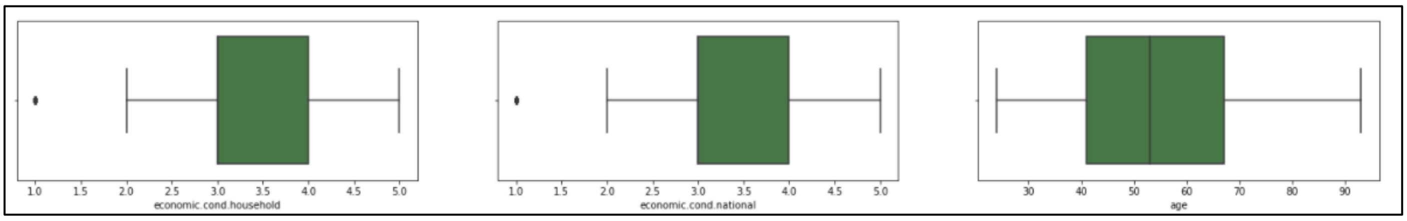
	count	unique	top	freq
vote	1517	2	Labour	1057
economic.cond.national	1517	5	3	604
economic.cond.household	1517	5	3	645
Blair	1517	5	4	833
Hague	1517	5	2	617
Europe	1517	11	11	338
political.knowledge	1517	4	2	776
gender	1517	2	female	808

- Almost 70% of the votes are for Labour party. The data is imbalanced. SMOTE technique is applied to the given data and the results are compared with original data.
- 22% of the voters represent ‘Eurosceptic’ sentiment.
- For Tony Blair-the Labour party leader, the frequency of 4 ratings is high.
- For William Hague-the conservative party leader, the frequency of 2 ratings is high.
- Almost 40-42% of voters have expressed neutral opinions about the current national and household economic conditions.

Outliers Treatment:

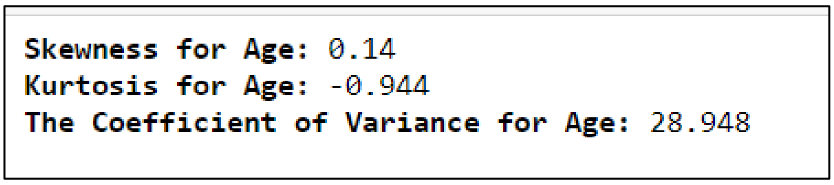
1. Very less people have assessed the current national economic and household economic condition less than 2.
2. Ratings less than 2 are shown as outliers.
3. As these are categories and not a wrong entry the outliers are not removed. Treating outliers in Target variable – Price will affect the predictions. No outlier treatment for Price variable.

Fig 1.1: Boxplot representation before treating outliers and Anomalies:



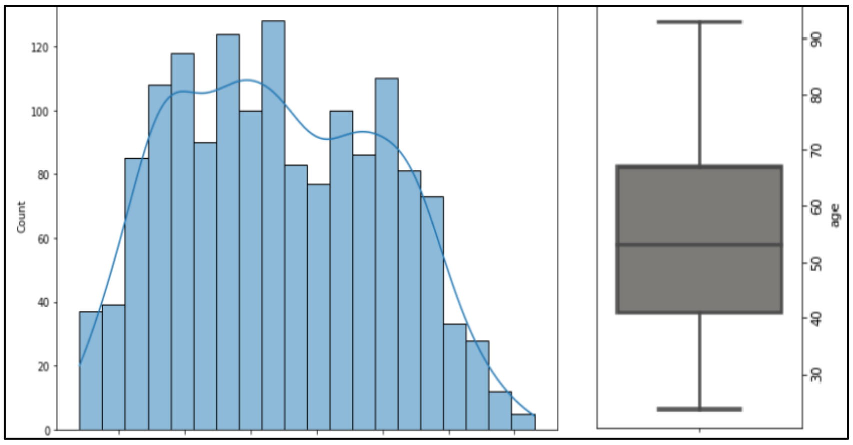
Exploratory Data Analysis:

Table 1.3: Skewness, Kurtosis and Coefficient of Variance (CV%) for Age column:



```
Skewness for Age: 0.14
Kurtosis for Age: -0.944
The Coefficient of Variance for Age: 28.948
```

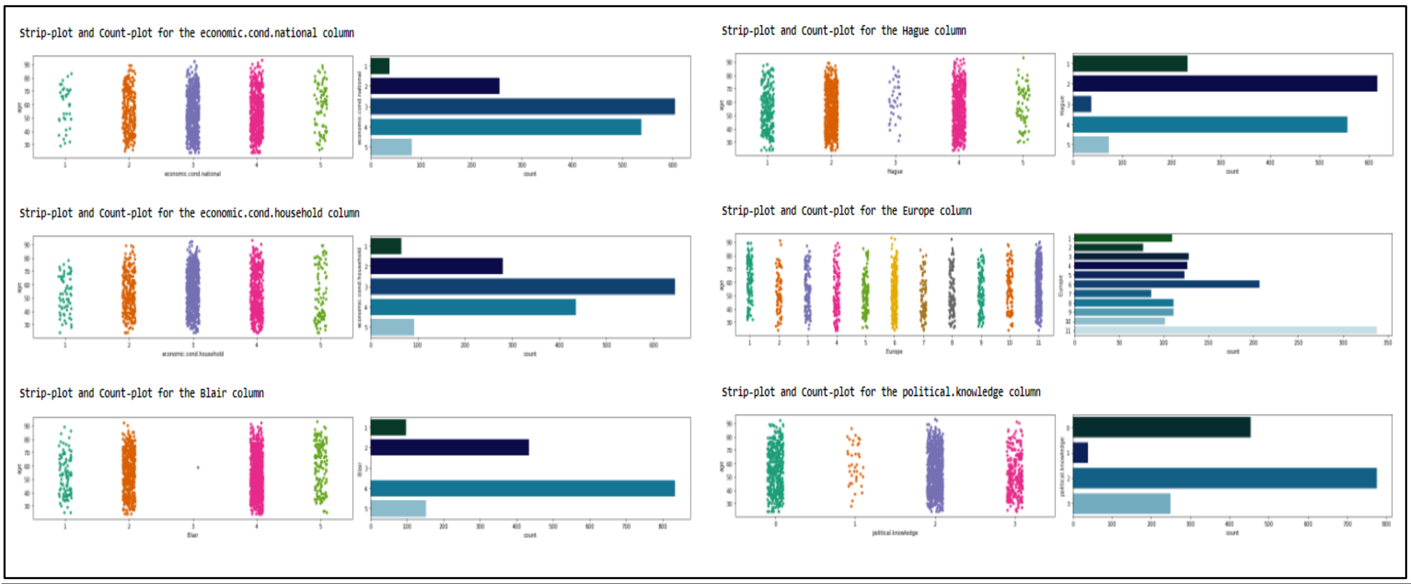
Fig 1.2: Univariate Analysis: Numerical columns:

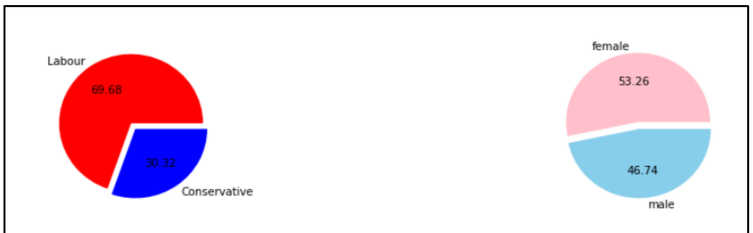


- The skewness for the age variable is 0.14 and it is symmetrical.
- The kurtosis for the age is very less. Which means no outliers and the distribution is flat and has thin tails
- the coefficient of variance is 29%. Lesser the CV% the feature has better stability.

1.2 Perform Univariate and Bivariate Analysis. Do exploratory data analysis. Check for Outliers. (7 Marks)

Fig 1.3: Univariate Analysis: Discrete columns:





Pie plot representation for Vote and Gender feature

Univariate analysis interpretation:

1. The mean age for the people voted for Labour is 53 and the mean age for the people who voted for conservative party is 56.

2. 75% of the people who have voted are aged below 70.

3. The target variable is found to be imbalanced. For the given problem SMOTE technique is used to over sample the minority class and results are compared with the original data.

4. The assessment for national economic conditions and household conditions follow similar pattern. Most common ratings are 2,3 and 4.

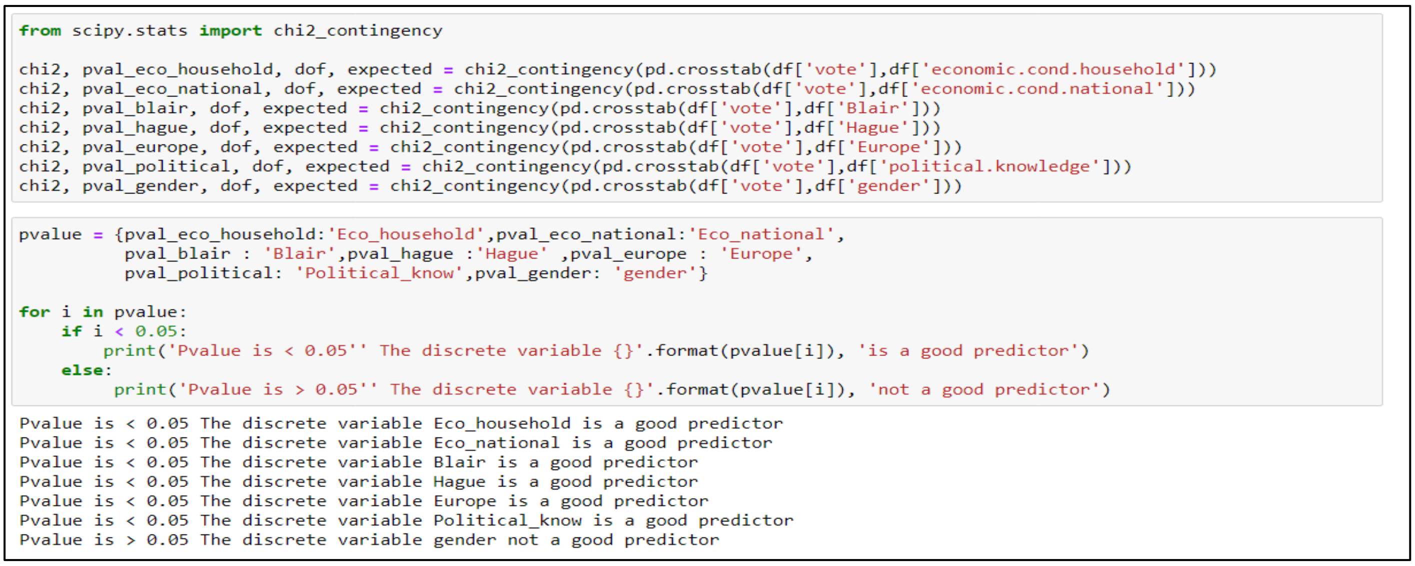
5. The assessment of leaders also have a similar pattern. Voters did not express Neutral opinions. The voters have strongly expressed their opinions towards leaders.

6. 22% of the voters have 'Eurosceptic' sentiment. 13% of the voters showed neutral opinion towards European integration.

7. Almost equal number of male and female voters are found.

Bivariate Analysis:

Fig 1.4: Bivariate Analysis: Chi Square contingency test to measure the correlation b/w Discrete variables



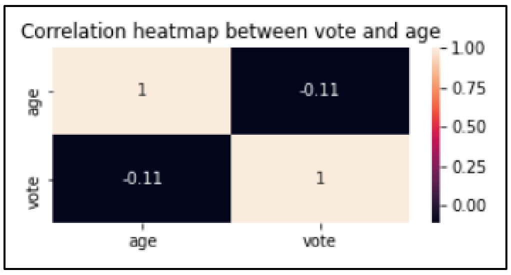
```
from scipy.stats import chi2_contingency

chi2, pval_eco_household, dof, expected = chi2_contingency(pd.crosstab(df['vote'],df['economic.cond.household']))
chi2, pval_eco_national, dof, expected = chi2_contingency(pd.crosstab(df['vote'],df['economic.cond.national']))
chi2, pval_blair, dof, expected = chi2_contingency(pd.crosstab(df['vote'],df['Blair']))
chi2, pval_hague, dof, expected = chi2_contingency(pd.crosstab(df['vote'],df['Hague']))
chi2, pval_europe, dof, expected = chi2_contingency(pd.crosstab(df['vote'],df['Europe']))
chi2, pval_political, dof, expected = chi2_contingency(pd.crosstab(df['vote'],df['political.knowledge']))
chi2, pval_gender, dof, expected = chi2_contingency(pd.crosstab(df['vote'],df['gender']))
```

```
pvalue = {pval_eco_household:'Eco_household',pval_eco_national:'Eco_national',
          pval_blair : 'Blair',pval_hague :'Hague' ,pval_europe : 'Europe',
          pval_political: 'Political_know',pval_gender: 'gender'}

for i in pvalue:
    if i < 0.05:
        print('Pvalue is < 0.05'' The discrete variable {}'.format(pvalue[i]), 'is a good predictor')
    else:
         print('Pvalue is > 0.05'' The discrete variable {}'.format(pvalue[i]), 'not a good predictor')
```

```
Pvalue is < 0.05 The discrete variable Eco_household is a good predictor
Pvalue is < 0.05 The discrete variable Eco_national is a good predictor
Pvalue is < 0.05 The discrete variable Blair is a good predictor
Pvalue is < 0.05 The discrete variable Hague is a good predictor
Pvalue is < 0.05 The discrete variable Europe is a good predictor
Pvalue is < 0.05 The discrete variable Political_know is a good predictor
Pvalue is > 0.05 The discrete variable gender not a good predictor
```

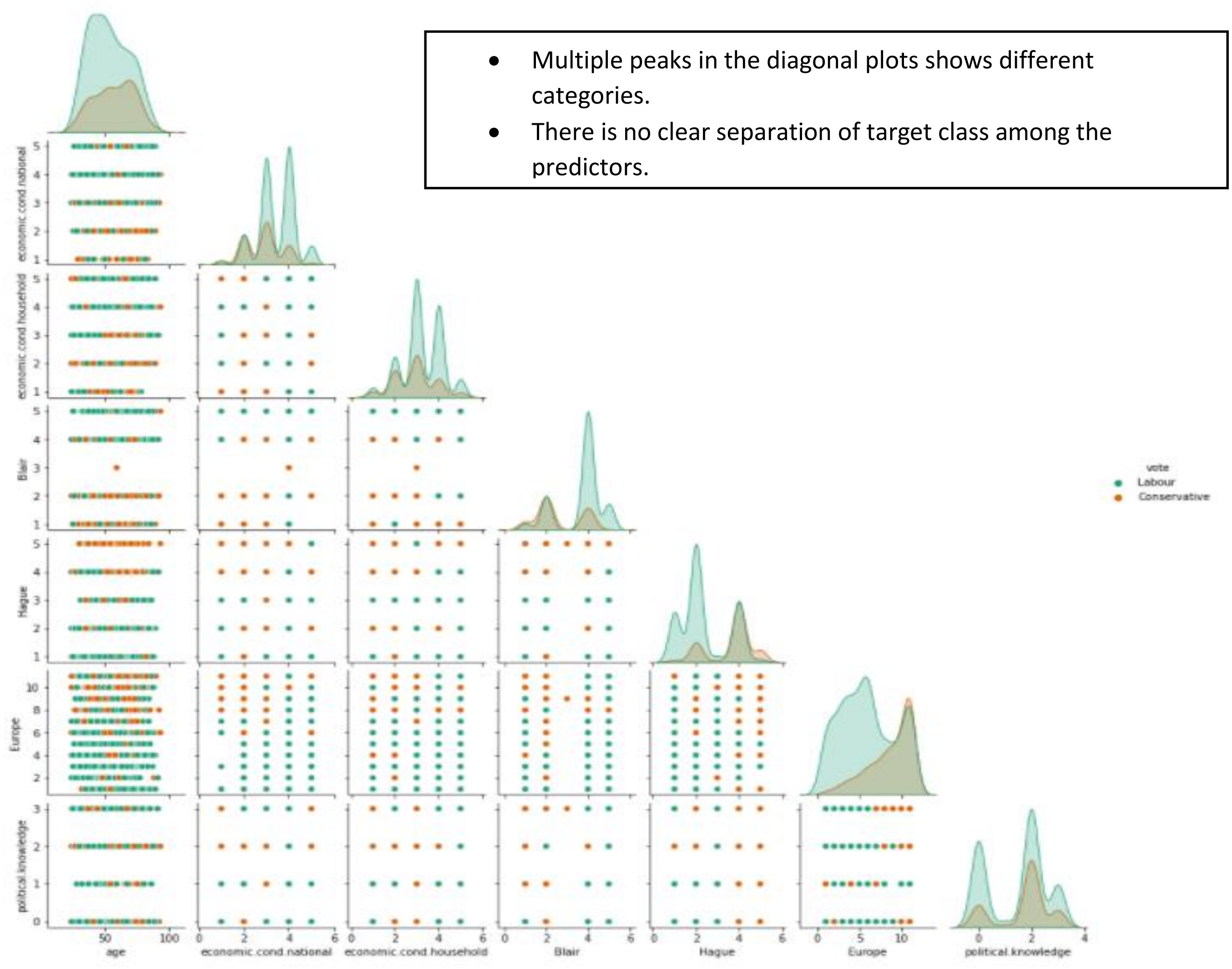
1. Assumption(H0): The two columns are NOT related to each other.
2. 5. If p value is less than 0.05, we reject the Null hypothesis and prove that the predictor variable is related to the target variable.
3. Except the variable 'gender' all other categorical variables are good predictors.
4. Relationship between predictors and target is required to have good output.
5. P-values for national economic condition, Hague, Blair and Europe are very less. Lesser the p-value stronger the relationship is.
6. The weights/importance of each variable can be checked using the coefficients from Logistic regression and LDA model.

Fig 1.4.1: Bivariate Analysis: Correlation plot between Vote and age:



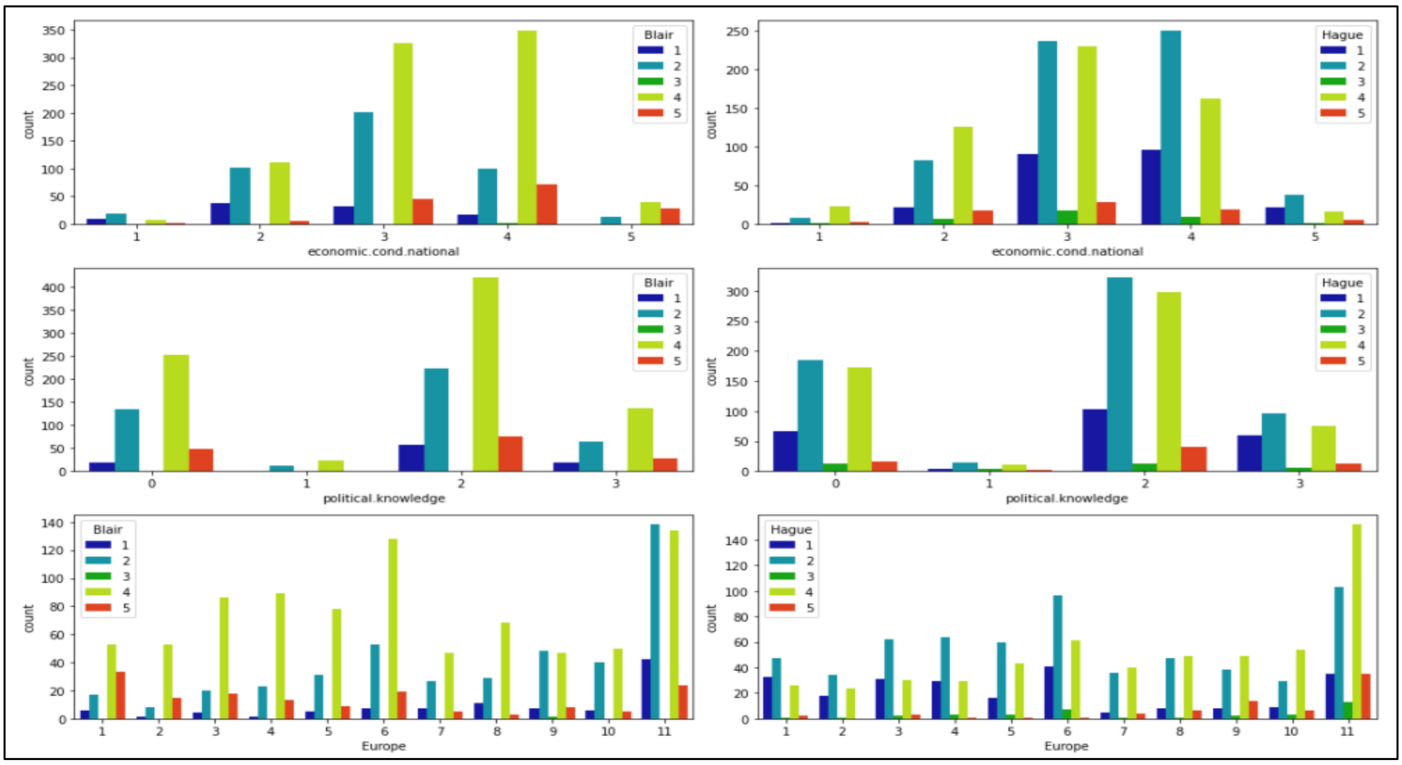
- There is a weak negative relationship between vote and age. Age is not a good predictor of vote.
- Willaim Hague, Tony Blair, national economic condition and European integration are the key predictors for predicting the vote.

Fig 1.5: Bivariate Analysis: Pair Plots



- Multiple peaks in the diagonal plots shows different categories.
- There is no clear separation of target class among the predictors.

Fig 1.6: Count-plot representation for categorical variables with Leader ratings as hue:



1. Voters who have rated Blair above 3 have expressed that there is good national economic condition. More number of voters have given 4 and above ratings.

2. More number of Voters who have rated Hague above 3 have expressed is neutral opinions towards national economic condition.

3. Blair: In national economic condition where the rating is more than 3, larger proportion of voters have rated Blair more than

4. Hague: In national economic condition where the rating is more than 3, larger proportion of voters have given 2 ratings to Hague.

5. Voters believe that Blair has stronger political knowledge on European integration.

6. Voters with high Eurosceptic attitude (11) prefer Hague.

* In the 5th subplot, from 1 to 6 i.e., voters who show negative attitude towards European integration have rated Blair above 4 frequently. From 6 to 11, there are high number of 2-ratings for Blair.

* In the 6th subplot, from 1 to 6 i.e., voters who show negative attitude towards European integration, have given more number of 2 ratings to Hague. From 6 to 11 i.e., towards high Eurosceptic attitude, ratings for Hague are in increasing trend.

Fig 1.6.1: Count-plot representation for categorical variables with Target variable:

1. Voters of Labour party have high ratings for assessment of current national economic and household economic.

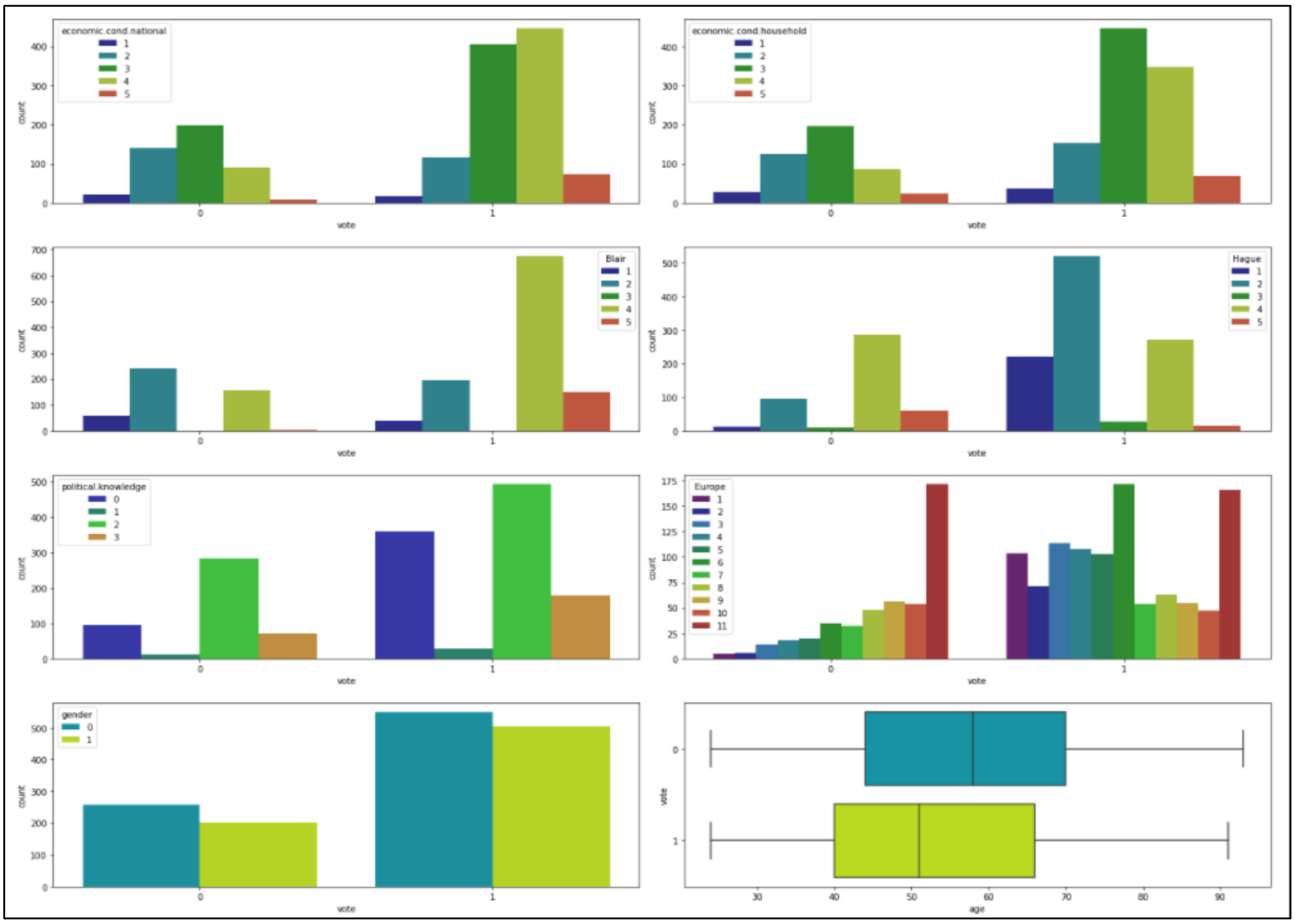
2. High ratings for Blair are from Voters of Labour party.

3. Similar trend between Class 1 and 0 is found for Political knowledge feature. In both the classes, more number of voters have rated their leaders 2 in a scale of 0 to 3.

4. The voters of conservative party are more Eurosceptic. Whereas voters of Blair have more neutral opinions along with high Eurosceptic attitude.

5. No clear discrimination between genders in selection of party.

6. The median age is little higher for the voters of conservative party.



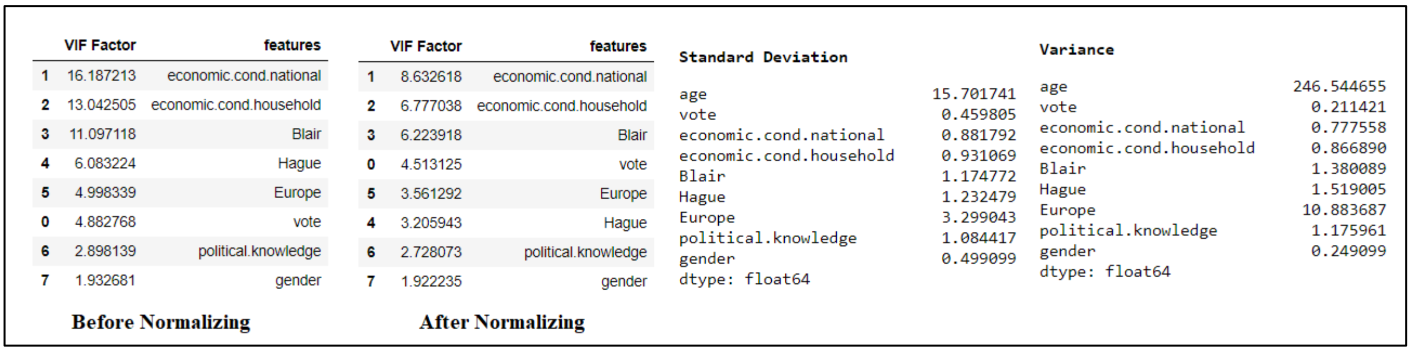
Key Predictors identified from Data Visualization:

Hague, Blair, Europe, Political Knowledge and National economic conditions

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). (4 Marks)

1. Except age all other columns are categorical. Age column will get higher weightage.
2. Europe feature has 11 labels. Normalizing the data using Min Max Scaler from Sklearn will bring all the data points within 0-1 range.
3. The VIF scores on unscaled data is more than 10, which means high collinearity.
4. The VIF scores of normalized data are less than 10. The collinearity is reduced.
5. The standard deviation and variance for Age is higher than other variables.
6. Hence, normalizing the data using Min Max scaler will improve the model performance.

Table 1.5: Variance Inflation Factors, Standard deviation and Variance



	VIF Factor	features
1	16.187213	economic.cond.national
2	13.042505	economic.cond.household
3	11.097118	Blair
4	6.083224	Hague
5	4.998339	Europe
0	4.882768	vote
6	2.898139	political.knowledge
7	1.932681	gender

Before Normalizing

	VIF Factor	features
1	8.632618	economic.cond.national
2	6.777038	economic.cond.household
3	6.223918	Blair
0	4.513125	vote
5	3.561292	Europe
4	3.205943	Hague
6	2.728073	political.knowledge
7	1.922235	gender

After Normalizing

```
Standard Deviation

age                        15.701741
vote                        0.459805
economic.cond.national      0.881792
economic.cond.household     0.931069
Blair                       1.174772
Hague                       1.232479
Europe                      3.299043
political.knowledge         1.084417
gender                      0.499099
dtype: float64
```

```
Variance

age                        246.544655
vote                         0.211421
economic.cond.national       0.777558
economic.cond.household      0.866890
Blair                        1.380089
Hague                        1.519005
Europe                      10.883687
political.knowledge          1.175961
gender                       0.249099
dtype: float64
```

Model Pre- processing:

Label Encoding:

Vote and gender columns are encoded as 0 and 1. 0 – Conservative and 1 – Labour. 0 – Female and 1- Male.

All other categorical columns are proceeded with the same labels.

Data Split:

1. The given data set is separated as X and y. X has all the predictor variables and y has the target variable.
2. X and y are split into 70:30 proportion with a random state as 1.
3. This **stratify** parameter makes a split so that the proportion of values in the sample produced will be the same as the proportion of values provided to parameter stratify.
4. Number of observations in training data: 1061
5. Number of observations in testing data: 456

1.4 Apply Logistic Regression and LDA (linear discriminant analysis). (4 marks)

1.5 Apply KNN Model and Naïve Bayes Model. Interpret the results. (4 marks)

Brief about the different models used:

Logistic Regression: Logistic regression is a linear model for classification rather than regression. It is also known as logit regression. In this model, the probabilities describing the possible outcomes of a single trial are modelled using a logistic function. The assumptions for Logistic regression are satisfied.

1. The data has low multi-collinearity.
2. There is a linear relationship between predictors and target.
3. Target variable is categorical and it is coded as 0 and 1.

Linear Discriminant analysis: LDA uses the linear combinations of variables to predict the class in the response variable of a given observation. LDA assumes that the independent variables are normally distributed and has equal variance.

Naïve Bayes Classifier: Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable.

K-Nearest neighbour classifier: Neighbours-based classification is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbours of each point: a query point is assigned the data class which has the most representatives within the nearest neighbours of the point.

Ensemble technique - Bagging Classifier: A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.

Ensemble technique- Ada Boosting Classifier: The core principle of AdaBoost is to fit a sequence of weak learners (i.e., models that are only slightly better than random guessing, such as small decision trees) on repeatedly modified versions of the data. The predictions from all of them are then combined through a weighted majority vote (or sum) to produce the final prediction.

Ensemble technique- Gradient Boosting Classifier: Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

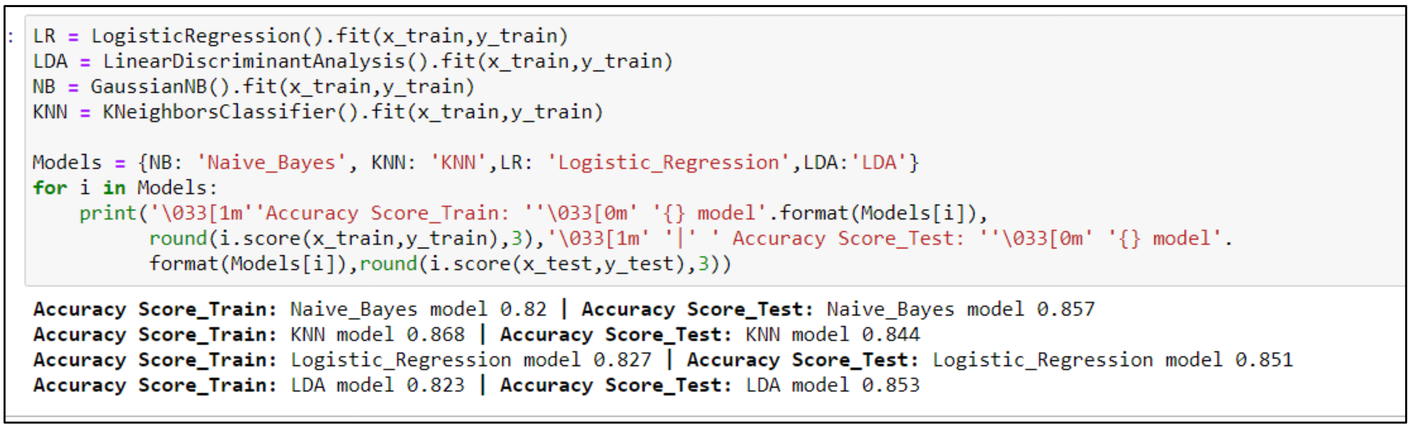
Ensemble technique - Voting Classifier: A Voting Classifier is a machine learning model that trains on an ensemble of numerous models and predicts an output (class) based on their highest probability of chosen class as the output. It simply aggregates the findings of each classifier passed into Voting Classifier and predicts the output class based on the highest majority of voting.

Note: Two different models for each classifier are created. One without using grid search and the latter using grid search function the hyper parameters are tuned. Important metrics are compared and the final best/optimized model is decided.

Performance metrics of base model (without tuning the hyperparameters):

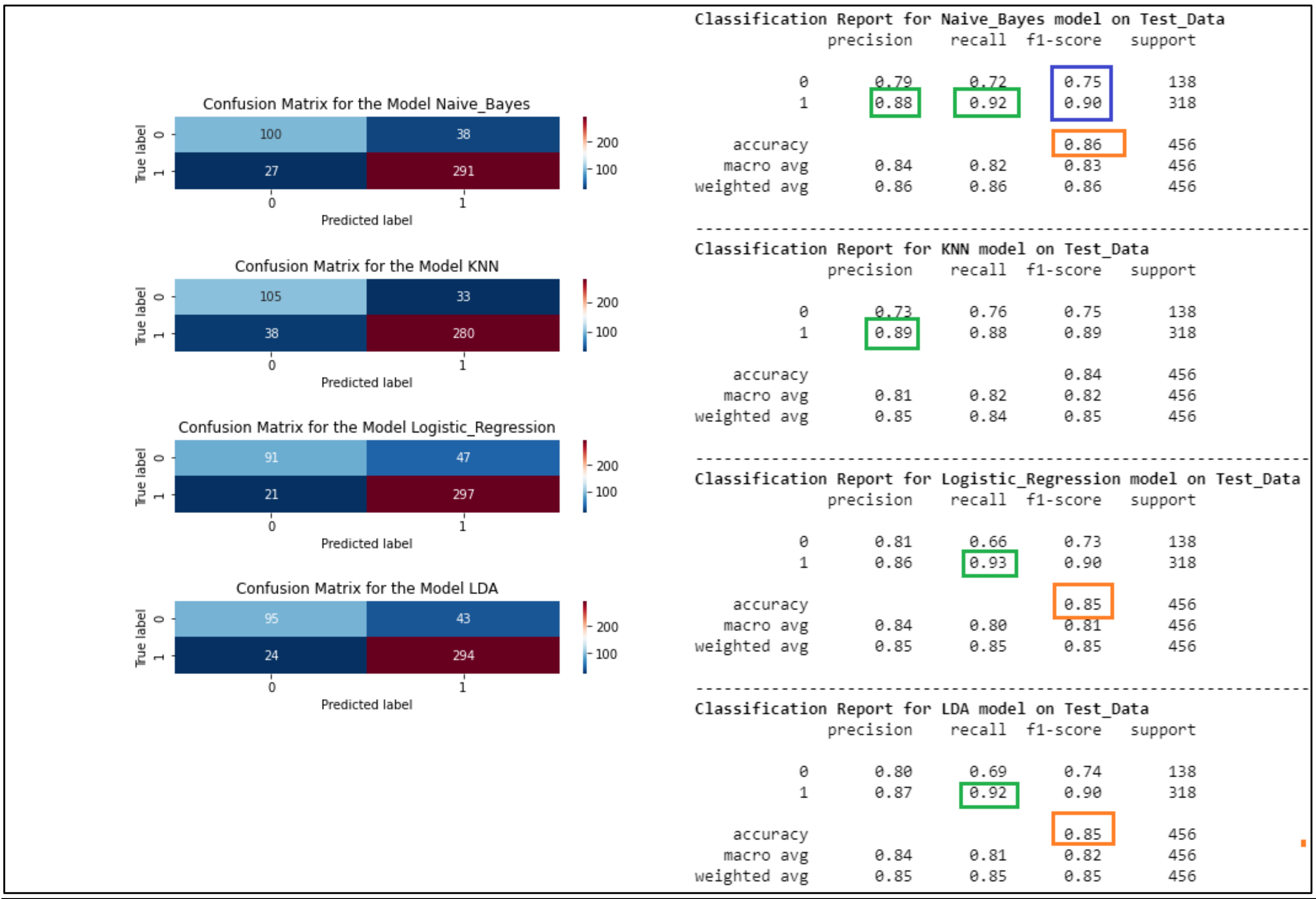
On the training and testing data, Logistic regression, LDA, Naïve bayes and K nearest neighbors are executed. Below are the accuracy scores on train and test.

Fig: 1.7.1 Accuracy scores on Train and Test on Models w/o model tuning:



```
LR = LogisticRegression().fit(x_train,y_train)
LDA = LinearDiscriminantAnalysis().fit(x_train,y_train)
NB = GaussianNB().fit(x_train,y_train)
KNN = KNeighborsClassifier().fit(x_train,y_train)

Models = {NB: 'Naive_Bayes', KNN: 'KNN',LR: 'Logistic_Regression',LDA:'LDA'}
for i in Models:
    print('\033[1m''Accuracy Score_Train: ''\033[0m' '{} model'.format(Models[i]),
          round(i.score(x_train,y_train),3),'\033[1m' '|' ' Accuracy Score_Test: ''\033[0m' '{} model'.
          format(Models[i]),round(i.score(x_test,y_test),3))

Accuracy Score_Train: Naive_Bayes model 0.82 | Accuracy Score_Test: Naive_Bayes model 0.857
Accuracy Score_Train: KNN model 0.868 | Accuracy Score_Test: KNN model 0.844
Accuracy Score_Train: Logistic_Regression model 0.827 | Accuracy Score_Test: Logistic_Regression model 0.851
Accuracy Score_Train: LDA model 0.823 | Accuracy Score_Test: LDA model 0.853
```

Fig: 1.7.2 Performance metrics for Test data on Model w/o model tuning:



```
Classification Report for Naive_Bayes model on Test_Data
              precision    recall  f1-score   support

           0       0.79      0.72      0.75       138
           1       0.88      0.92      0.90       318

    accuracy                           0.86       456
   macro avg       0.84      0.82      0.83       456
weighted avg       0.86      0.86      0.86       456

---------------------------------------------------------------------
Classification Report for KNN model on Test_Data
              precision    recall  f1-score   support

           0       0.73      0.76      0.75       138
           1       0.89      0.88      0.89       318

    accuracy                           0.84       456
   macro avg       0.81      0.82      0.82       456
weighted avg       0.85      0.84      0.85       456

---------------------------------------------------------------------
Classification Report for Logistic_Regression model on Test_Data
              precision    recall  f1-score   support

           0       0.81      0.66      0.73       138
           1       0.86      0.93      0.90       318

    accuracy                           0.85       456
   macro avg       0.84      0.80      0.81       456
weighted avg       0.85      0.85      0.85       456

---------------------------------------------------------------------
Classification Report for LDA model on Test_Data
              precision    recall  f1-score   support

           0       0.80      0.69      0.74       138
           1       0.87      0.92      0.90       318

    accuracy                           0.85       456
   macro avg       0.84      0.81      0.82       456
weighted avg       0.85      0.85      0.85       456
```

1. *True Positives: Actual Labour-Predicted Labour, True Negatives: Actual Conservative-Predicted Conservative, False Negative: Actual Labour-Predicted Conservative, False Positive: Actual Conservative-Predicted Labour*.

2. Naïve bayes, Logistic regression and LDA models have similar accuracies.

3. No overfitting/underfitting in the model performance. i.e., difference between the train and test accuracy scores not more than 10%.

4. Recall: Actual true data points identified as true data points by the model. Naïve bayes, Logistic regression and LDA models have good recall scores for Labour class. As the data is imbalanced the recall percentage difference between 0 and 1 are huge.

5. Precision: Among the identified true points how many are really positive. KNN and Naïve bayes shows good precision scores.

6. For the given problem statement, *Recall and Precision* both are equally important. A voter actually voted for Labour party predicted as conservative and vice versa, both will impact the exit poll which will impact the overall win and seats covered.

7. Hence, a model which gives almost equal score for both recall and precision should be considered.

8. The recall and precision score for KNN model is equal. But the accuracy is low when compared with other models.

9. Let's check the accuracy scores and other metrics by tuning the hyper parameters for the above models. Ensemble methods like Bagging, Boosting and Voting are also executed for the given problem. The final best model will be selected on comparing the metrics between the different models.

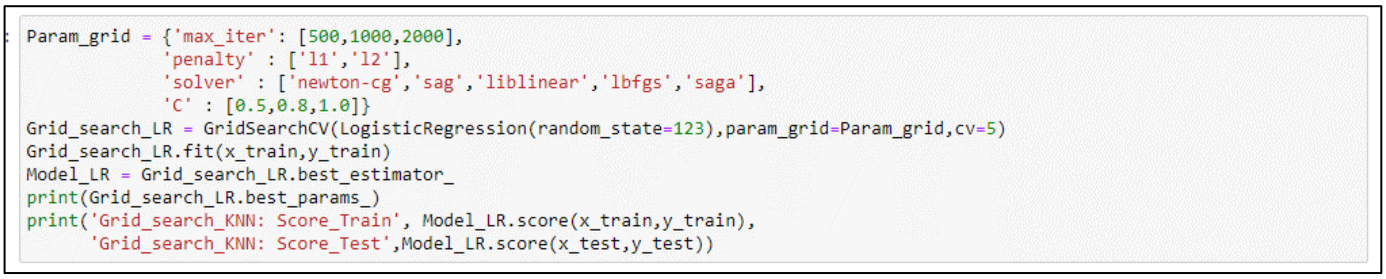
1.6 Model Tuning, Bagging (Random Forest should be applied for Bagging), and Boosting. (7 marks)

Model Tuning, Bagging, Boosting and Voting

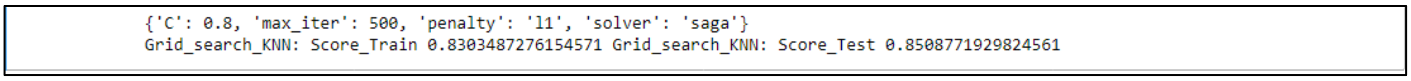
Naïve Bayes Model: No hyper parameters to tune. The original model is considered for comparison.

Logistic Regression:

Fig: 1.8.1 Logistic Regression using Grid Search:


```
Param_grid = {'max_iter': [500,1000,2000],
              'penalty' : ['l1','l2'],
              'solver' : ['newton-cg','sag','liblinear','lbfgs','saga'],
              'C' : [0.5,0.8,1.0]}
Grid_search_LR = GridSearchCV(LogisticRegression(random_state=123),param_grid=Param_grid,cv=5)
Grid_search_LR.fit(x_train,y_train)
Model_LR = Grid_search_LR.best_estimator_
print(Grid_search_LR.best_params_)
print('Grid_search_KNN: Score_Train', Model_LR.score(x_train,y_train),
      'Grid_search_KNN: Score_Test',Model_LR.score(x_test,y_test))
```

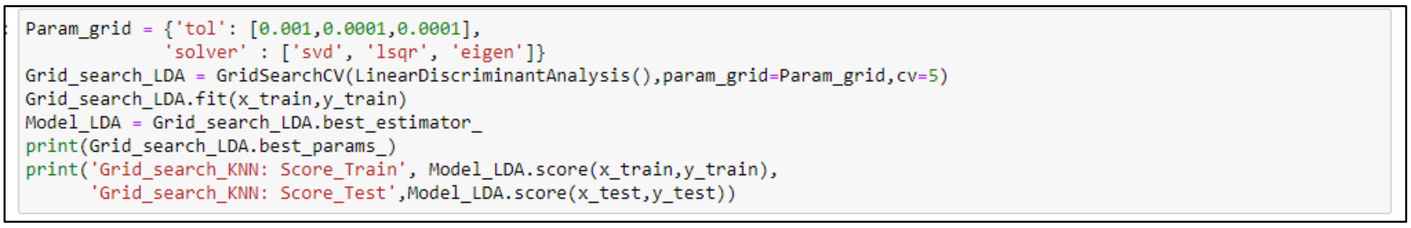
- Max_iterations: Maximum number of iterations of the optimization algorithm. Default is 100. The model here is tuned with [500,1000,2000].
- Solver: For small datasets, 'liblinear' is preferred. 'sag' and 'saga' are faster for larger ones. Among the given different solvers, model has chosen 'saga'. Sag and Saga both has fast convergence on features with same scale. As we have normalized the data, Logit model with solver 'saga' gives good accuracy.
- Penalty: Used to specify the norm used in the penalization. The 'newton-cg', 'sag' and 'lbfgs' solvers support only l2 penalties.
- C: Each of the values in Cs describes the inverse of regularization strength. Smaller C's specifies stronger regularization.
- CV: cross-validation generator. For the given model, we have used 5 stratified folds. 4 for train and 1 for test.
- Best params on which the model is executed:


```
{'C': 0.8, 'max_iter': 500, 'penalty': 'l1', 'solver': 'saga'}
Grid_search_KNN: Score_Train 0.8303487276154571 Grid_search_KNN: Score_Test 0.8508771929824561
```

- The accuracy for the model has no changes.

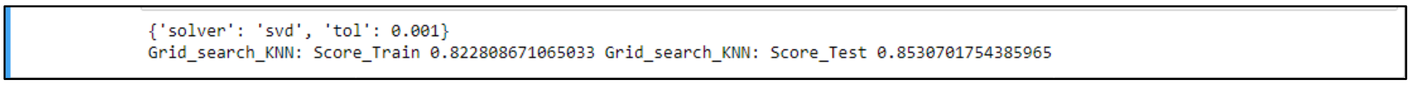
Linear discriminant analysis:

Fig: 1.8.2 LDA using Grid Search:



```
Param_grid = {'tol': [0.001,0.0001,0.0001],
             'solver' : ['svd', 'lsqr', 'eigen']}
Grid_search_LDA = GridSearchCV(LinearDiscriminantAnalysis(),param_grid=Param_grid,cv=5)
Grid_search_LDA.fit(x_train,y_train)
Model_LDA = Grid_search_LDA.best_estimator_
print(Grid_search_LDA.best_params_)
print('Grid_search_KNN: Score_Train', Model_LDA.score(x_train,y_train),
      'Grid_search_KNN: Score_Test',Model_LDA.score(x_test,y_test))
```

1. Tol: Threshold used for rank estimation in SVD solver.
2. Solvers: ‘svd’: Singular value decomposition (default). Does not compute the covariance matrix, therefore this solver is recommended for data with a large number of features. ‘lsqr’: Least squares solution, can be combined with shrinkage. ‘eigen’: Eigenvalue decomposition, can be combined with shrinkage.
3. Best params on which the model is executed:



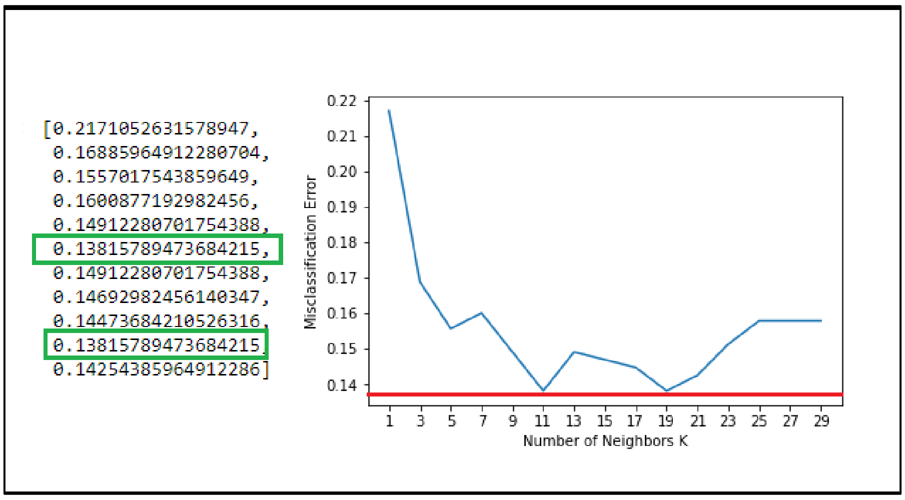
```
{'solver': 'svd', 'tol': 0.001}
Grid_search_KNN: Score_Train 0.822808671065033 Grid_search_KNN: Score_Test 0.8530701754385965
```

4. The accuracy scores show no change after tuning the hyper parameters.

K-Nearest neighbors:

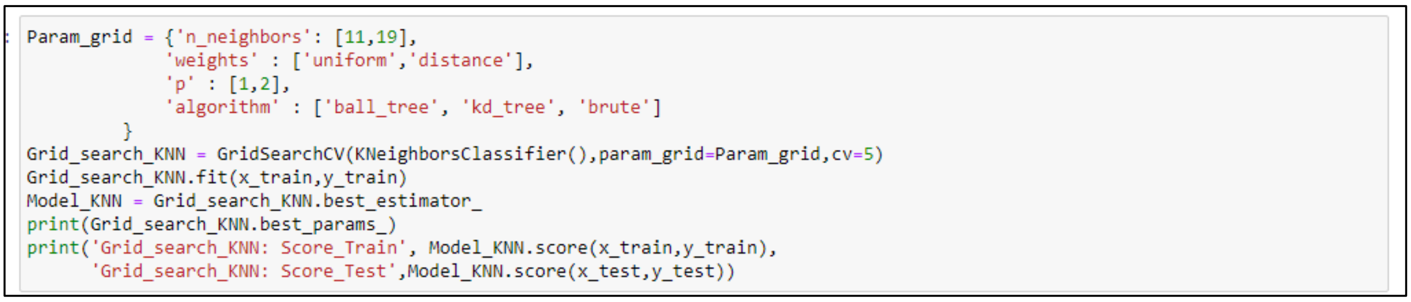
1. To find right value for k, scores of k ranging from 1 to 30 are analysed. The misclassification error is minimal when k=11 and 19. The below graph, plots the misclassification errors from different k’s. Misclassification errors clearly increase from 19.

Fig: 1.8.3 Misclassification plot for k = (1 to 30)

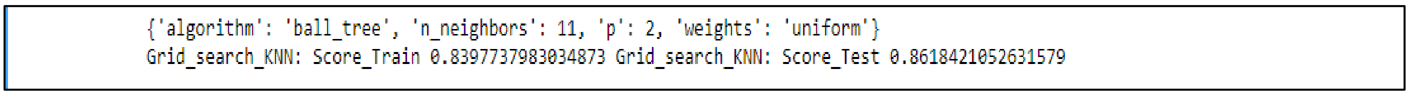


```
[0.2171052631578947,
 0.16885964912280704,
 0.1557017543859649,
 0.1600877192982456,
 0.1491228070175438,
 0.13815789473684215,
 0.1491228070175438,
 0.14692982456140347,
 0.14473684210526316,
 0.13815789473684215,
 0.14254385964912286]
```

Fig: 1.8.4 KNN using Grid Search:


```
Param_grid = {'n_neighbors': [11,19],
              'weights' : ['uniform','distance'],
              'p' : [1,2],
              'algorithm' : ['ball_tree', 'kd_tree', 'brute']
          }
Grid_search_KNN = GridSearchCV(KNeighborsClassifier(),param_grid=Param_grid,cv=5)
Grid_search_KNN.fit(x_train,y_train)
Model_KNN = Grid_search_KNN.best_estimator_
print(Grid_search_KNN.best_params_)
print('Grid_search_KNN: Score_Train', Model_KNN.score(x_train,y_train),
      'Grid_search_KNN: Score_Test',Model_KNN.score(x_test,y_test))
```

1. N_neighbors: 11 and 19. The given k value is decided by looking at the misclassification graphical representation.
2. Weights: weight function used in prediction. 'uniform': uniform weights. All points in each neighbourhood are weighted equally. 'distance': weight points by the inverse of their distance. In this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
3. P: Power parameter for the Minkowski metric. When p = 1, this is equivalent to using manhattan_distance (l1), and euclidean_distance (l2) for p = 2.
4. Algorithm: Different algorithms used here are 'ball-tree', 'kd_tree','brute'.
5. Best params on which the model is executed:


```
{'algorithm': 'ball_tree', 'n_neighbors': 11, 'p': 2, 'weights': 'uniform'}
Grid_search_KNN: Score_Train 0.8397737983034873 Grid_search_KNN: Score_Test 0.8618421052631579
```

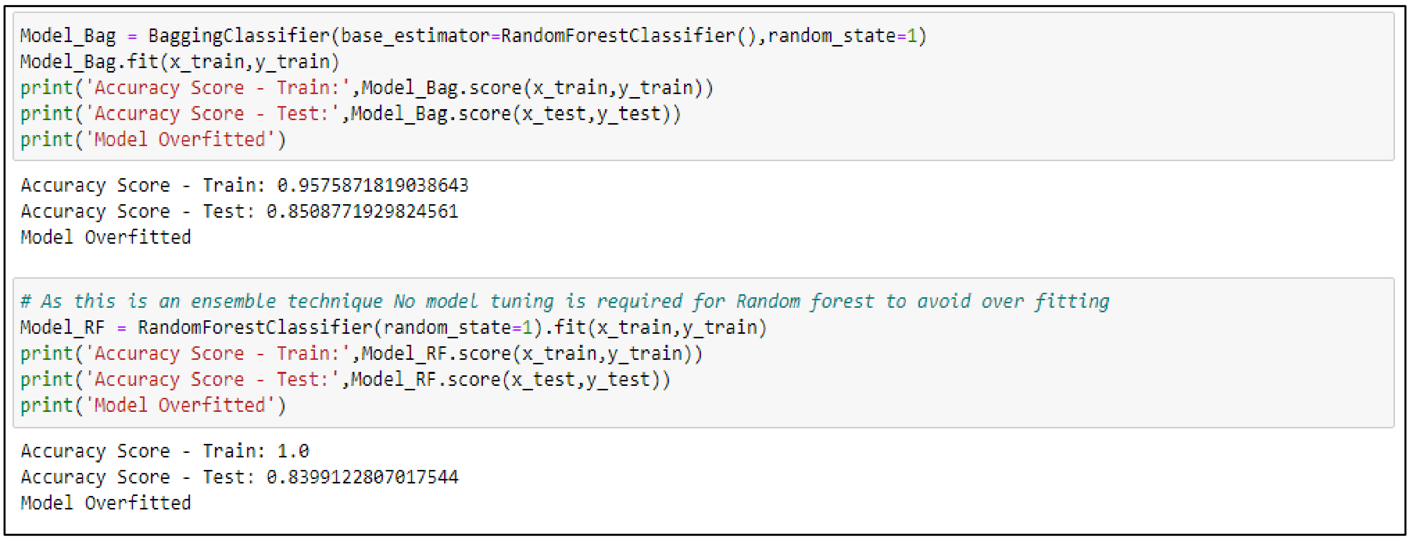
6. The accuracy scores have improved from 84% to 86%

Ensemble Methods:

Bagging Classifier and Random Forest Classifier:

1. The bagging classifier and the random forest classifier are executed without tuning the hyper parameters.
2. The difference between train and test accuracy scores are more than 10%. Both the model shows 'over-fitting'
3. Hence, the hyper parameters are tuned using Grid search to reduce the high variance.

Fig 1.8.5: Bagging and Random Scores without using Grid Search:



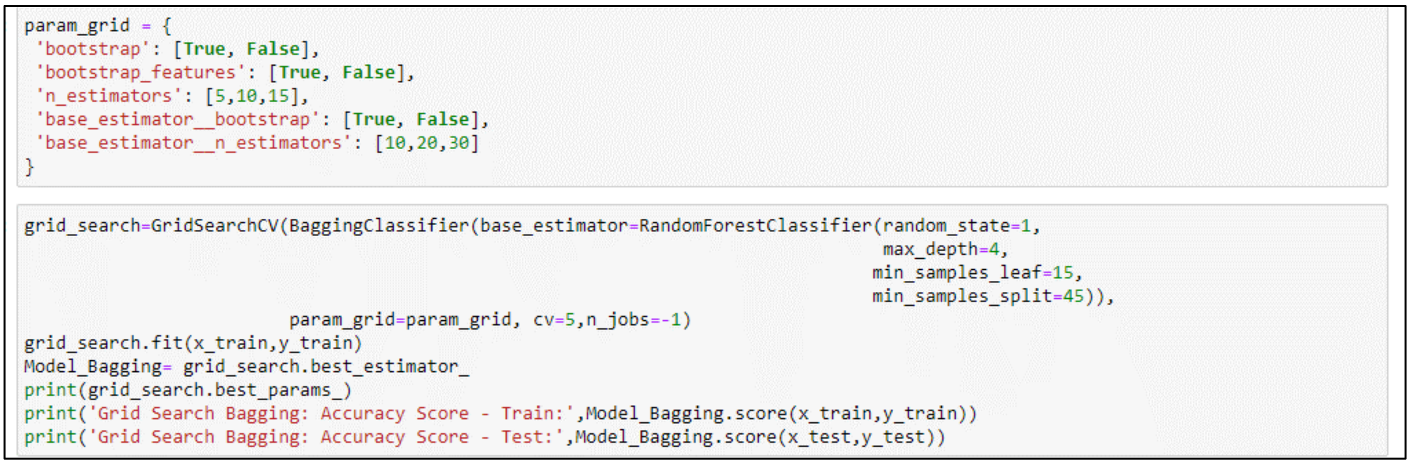
```
Model_Bag = BaggingClassifier(base_estimator=RandomForestClassifier(),random_state=1)
Model_Bag.fit(x_train,y_train)
print('Accuracy Score - Train:',Model_Bag.score(x_train,y_train))
print('Accuracy Score - Test:',Model_Bag.score(x_test,y_test))
print('Model Overfitted')
```

```
Accuracy Score - Train: 0.9575871819038643
Accuracy Score - Test: 0.8508771929824561
Model Overfitted
```

```
# As this is an ensemble technique No model tuning is required for Random forest to avoid over fitting
Model_RF = RandomForestClassifier(random_state=1).fit(x_train,y_train)
print('Accuracy Score - Train:',Model_RF.score(x_train,y_train))
print('Accuracy Score - Test:',Model_RF.score(x_test,y_test))
print('Model Overfitted')
```

```
Accuracy Score - Train: 1.0
Accuracy Score - Test: 0.8399122807017544
Model Overfitted
```

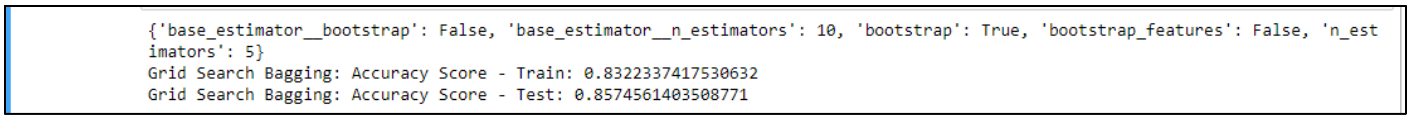
Fig 1.8.6: Grid Search for Bagging Classifier with Random Forest as base estimator:



```
param_grid = {
 'bootstrap': [True, False],
 'bootstrap_features': [True, False],
 'n_estimators': [5,10,15],
 'base_estimator__bootstrap': [True, False],
 'base_estimator__n_estimators': [10,20,30]
}
```

```
grid_search=GridSearchCV(BaggingClassifier(base_estimator=RandomForestClassifier(random_state=1,
                                                                                  max_depth=4,
                                                                                 min_samples_leaf=15,
                                                                                 min_samples_split=45)),
                         param_grid=param_grid, cv=5,n_jobs=-1)
grid_search.fit(x_train,y_train)
Model_Bagging= grid_search.best_estimator_
print(grid_search.best_params_)
print('Grid Search Bagging: Accuracy Score - Train:',Model_Bagging.score(x_train,y_train))
print('Grid Search Bagging: Accuracy Score - Test:',Model_Bagging.score(x_test,y_test))
```

1. Base estimator: the base estimator is Random Forest classifier.
2. N_estimator: The number of base estimators in the ensemble. [5,10,15]
3. Bootstrap: Whether samples are drawn with replacement.
4. bootstrap_features: Whether features are drawn with replacement.
5. Base_estimator_n_estimators: Number of estimators for random forest classifiers.
6. Best parameters on which the model built is.



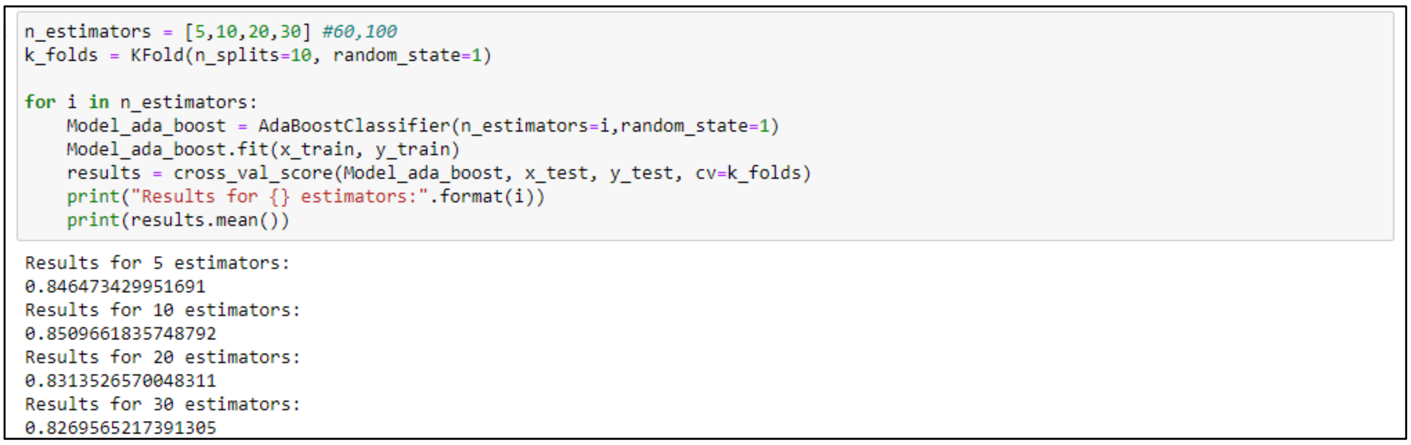
```
{'base_estimator__bootstrap': False, 'base_estimator__n_estimators': 10, 'bootstrap': True, 'bootstrap_features': False, 'n_est
imators': 5}
Grid Search Bagging: Accuracy Score - Train: 0.8322337417530632
Grid Search Bagging: Accuracy Score - Test: 0.8574561403508771
```

7. The bagging classifier and Random Forest before hyper parameters tuning showed 'over-fit'/high variance. After tuning the hyper parameters, the difference between the train and test scores are very minimal. *Over-fitting or high variance is removed by using grid search.*

Ada Boosting classifier:

An important hyperparameter for Ada boost is **n_estimators.** Often by changing the number of base models or weak learners we can adjust the accuracy of the model.

Fig 1.9: Number of estimators for Ada Boosting:

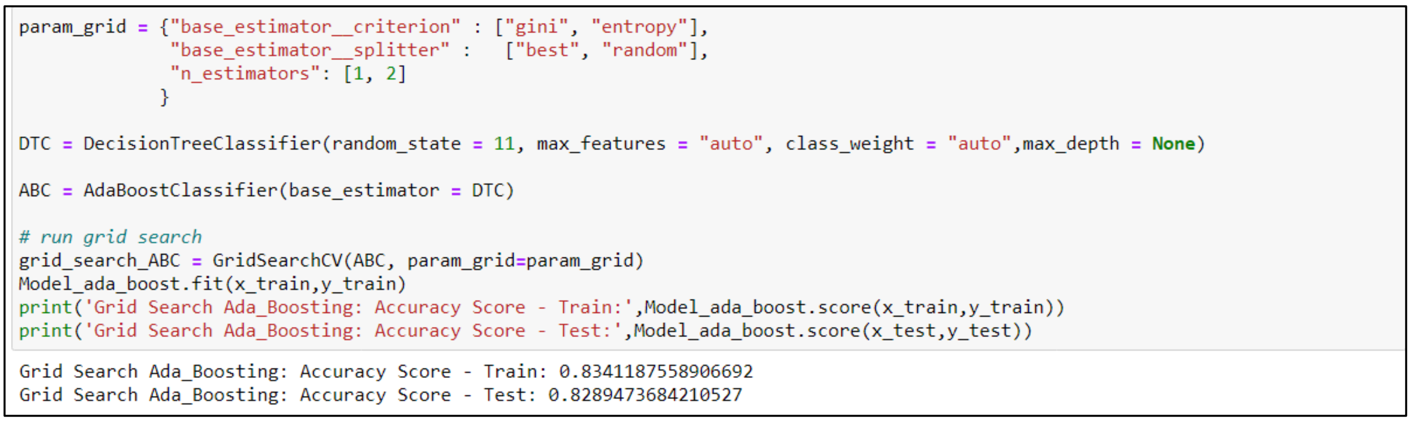


```
n_estimators = [5,10,20,30] #60,100
k_folds = KFold(n_splits=10, random_state=1)

for i in n_estimators:
    Model_ada_boost = AdaBoostClassifier(n_estimators=i,random_state=1)
    Model_ada_boost.fit(x_train, y_train)
    results = cross_val_score(Model_ada_boost, x_test, y_test, cv=k_folds)
    print("Results for {} estimators:".format(i))
    print(results.mean())
```

```
Results for 5 estimators:
0.846473429951691
Results for 10 estimators:
0.8509661835748792
Results for 20 estimators:
0.8313526570048311
Results for 30 estimators:
0.8269565217391305
```

- The above figure shows the optimum number of estimators to be chosen for the model. The accuracy score is good when the number of estimators is 10.

Fig 1.9.1: Grid Search CV for Ada boosting:

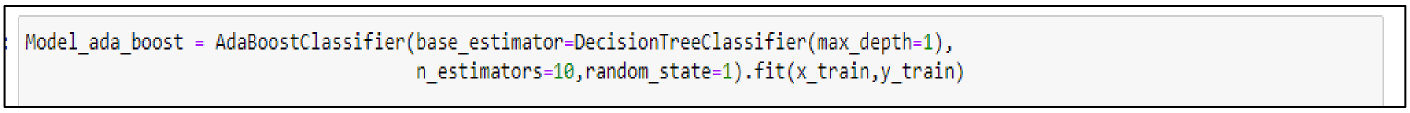


```
param_grid = {"base_estimator__criterion" : ["gini", "entropy"],
              "base_estimator__splitter" :   ["best", "random"],
              "n_estimators": [1, 2]
             }

DTC = DecisionTreeClassifier(random_state = 11, max_features = "auto", class_weight = "auto",max_depth = None)

ABC = AdaBoostClassifier(base_estimator = DTC)

# run grid search
grid_search_ABC = GridSearchCV(ABC, param_grid=param_grid)
Model_ada_boost.fit(x_train,y_train)
print('Grid Search Ada_Boosting: Accuracy Score - Train:',Model_ada_boost.score(x_train,y_train))
print('Grid Search Ada_Boosting: Accuracy Score - Test:',Model_ada_boost.score(x_test,y_test))
```

```
Grid Search Ada_Boosting: Accuracy Score - Train: 0.8341187558906692
Grid Search Ada_Boosting: Accuracy Score - Test: 0.8289473684210527
```

- The base estimator Decision tree classifier is tuned to fit the ada boosting model.
- The Maximum features and class_weight is given 'auto' option and no maximum depth is given.
- The accuracy score from Grid search CV for Ada boosting is 82.8%

Ada Boosting Model (Weak learner as base estimator):



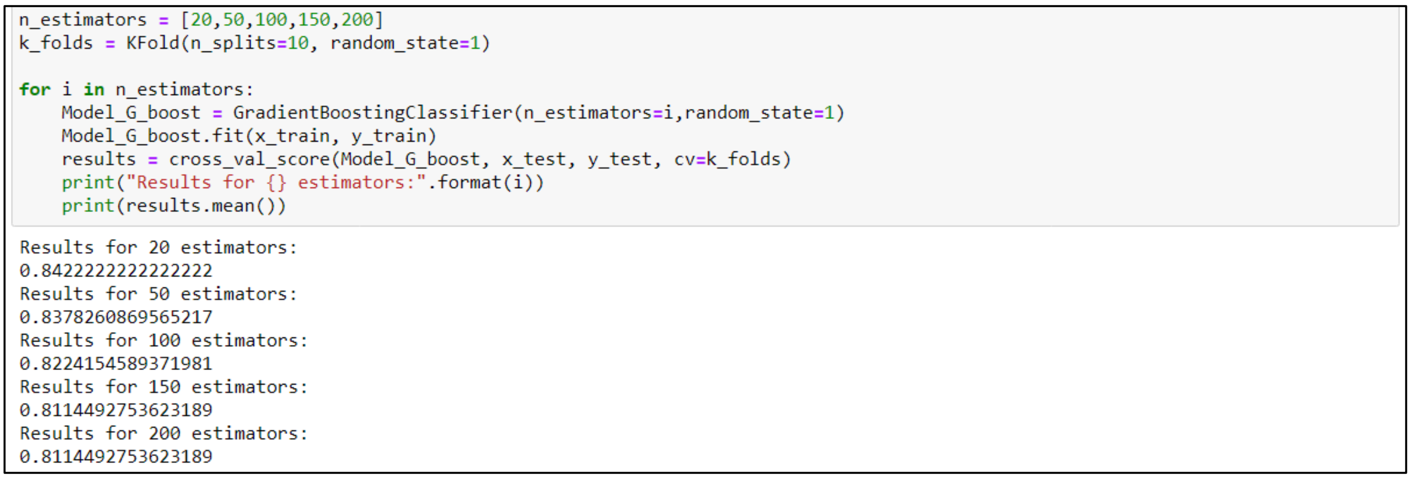
```
Model_ada_boost = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1),
                                     n_estimators=10,random_state=1).fit(x_train,y_train)
```

- Ada boosting model performs well by boosting the scores of the weak learners. As the accuracy scores are low with Grid search, the Ada boost model is executed with the default parameters (Decision tree with Maximum depth as 1). The accuracy score improved to 85.01%

Gradient Boosting classifier:

An important hyperparameter for Gradient boost is **n_estimators.** Often by changing the number of base models or weak learners we can adjust the accuracy of the model.

Fig 1.9.3: Number of estimators for Gradient Boosting:

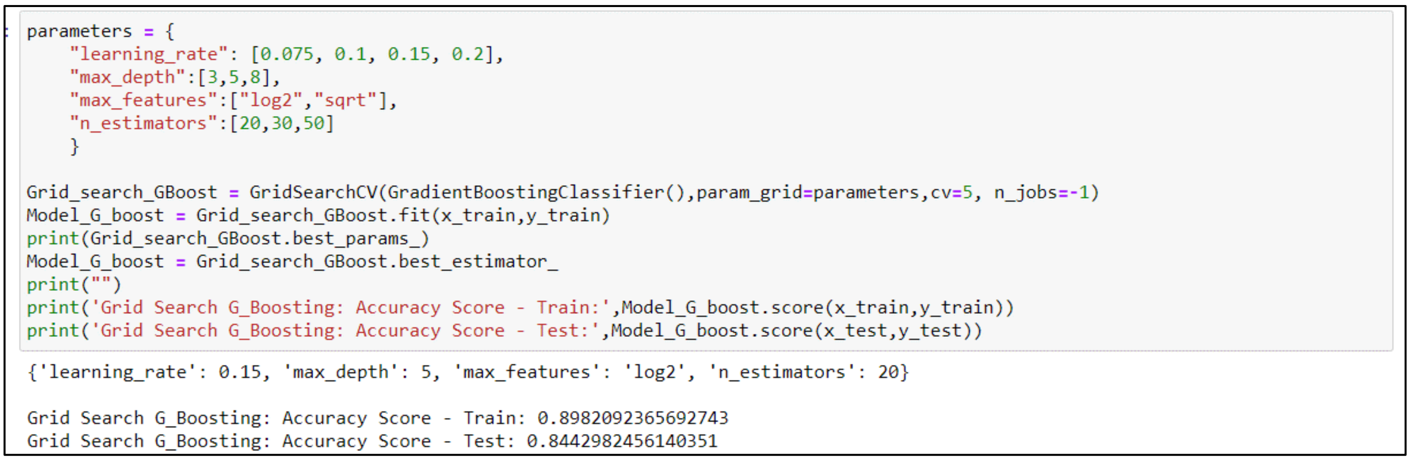


```
n_estimators = [20,50,100,150,200]
k_folds = KFold(n_splits=10, random_state=1)

for i in n_estimators:
    Model_G_boost = GradientBoostingClassifier(n_estimators=i,random_state=1)
    Model_G_boost.fit(x_train, y_train)
    results = cross_val_score(Model_G_boost, x_test, y_test, cv=k_folds)
    print("Results for {} estimators:".format(i))
    print(results.mean())
```

```
Results for 20 estimators:
0.8422222222222222
Results for 50 estimators:
0.8378260869565217
Results for 100 estimators:
0.8224154589371981
Results for 150 estimators:
0.8114492753623189
Results for 200 estimators:
0.8114492753623189
```

- The above figure shows the optimum number of estimators to be chosen for the model. The accuracy score is good when the number of estimators is 20.

Fig 1.9.3: Grid Search CV for Gradient boosting:



```
parameters = {
    "learning_rate": [0.075, 0.1, 0.15, 0.2],
    "max_depth":[3,5,8],
    "max_features":["log2","sqrt"],
    "n_estimators":[20,30,50]
    }

Grid_search_GBoost = GridSearchCV(GradientBoostingClassifier(),param_grid=parameters,cv=5, n_jobs=-1)
Model_G_boost = Grid_search_GBoost.fit(x_train,y_train)
print(Grid_search_GBoost.best_params_)
Model_G_boost = Grid_search_GBoost.best_estimator_
print("")
print('Grid Search G_Boosting: Accuracy Score - Train:',Model_G_boost.score(x_train,y_train))
print('Grid Search G_Boosting: Accuracy Score - Test:',Model_G_boost.score(x_test,y_test))
```

```
{'learning_rate': 0.15, 'max_depth': 5, 'max_features': 'log2', 'n_estimators': 20}

Grid Search G_Boosting: Accuracy Score - Train: 0.8982092365692743
Grid Search G_Boosting: Accuracy Score - Test: 0.8442982456140351
```

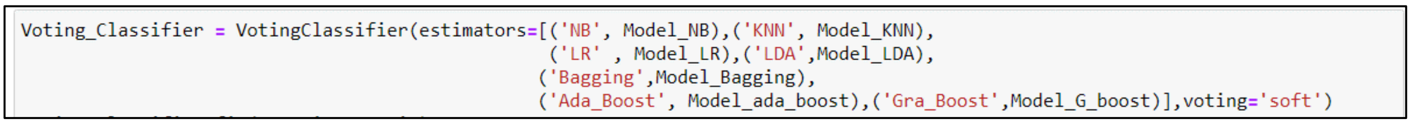
- Learning rate: the learning rate shrinks the contribution of each tree by the given learning rate.
- Max depth: this indicated the depth of the tree.
- N_estimators: Number of trees in the forest.
- Max_features: represents the number of features to consider when looking for the best split.

- The accuracy score from Gradient boosting using grid search is 84.42%. The difference between the train and test accuracies are less than 10%. No over fit.

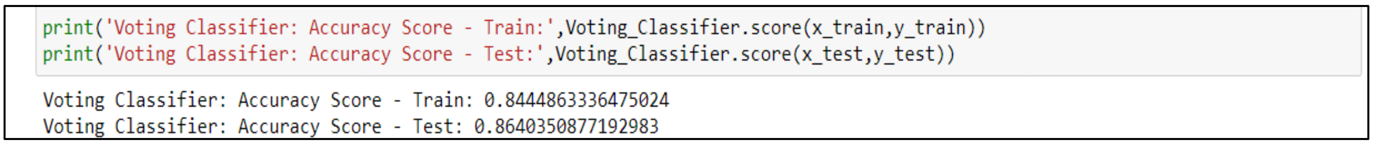
Voting Classifier:

The Bagging classifier is the ensemble learning of homogenous models whereas, Voting Classifier and predicts the output class based on the highest majority of voting from heterogenous models. For the given problem we have executed 7 different models. (NB, KNN, LR, LDA, Bagging, Ada Boosting and Gradient Boosting). Voting classifier gives prediction using the majority vote and also improves the accuracies.

Fig 1.9.4: Voting Classifier:



```
Voting_Classifier = VotingClassifier(estimators=[('NB', Model_NB),('KNN', Model_KNN),
                                                  ('LR' , Model_LR),('LDA',Model_LDA),
                                                 ('Bagging',Model_Bagging),
                                                 ('Ada_Boost', Model_ada_boost),('Gra_Boost',Model_G_boost)],voting='soft')
```



```
print('Voting Classifier: Accuracy Score - Train:',Voting_Classifier.score(x_train,y_train))
print('Voting Classifier: Accuracy Score - Test:',Voting_Classifier.score(x_test,y_test))

Voting Classifier: Accuracy Score - Train: 0.8444863336475024
Voting Classifier: Accuracy Score - Test: 0.8640350877192983
```

- The estimators in the voting classifier contains all the executed models.
- There are two types of voting: hard and soft. In hard voting, the predicted output class is a class with the highest majority of votes i.e., the class which had the highest probability of being predicted by each of the classifiers. In soft voting, the output class is the prediction based on the average of probability given to that class.
- The accuracy score is high when the voting type is ‘soft’.
- The accuracy from the Voting classifier is 86.40%

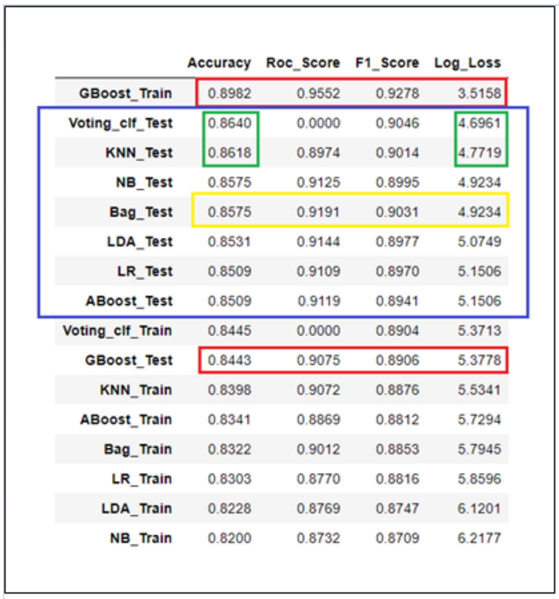
1.7 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model. Final Model: Compare the models and write inference which model is best/optimized. (7 marks)

SMOTE:

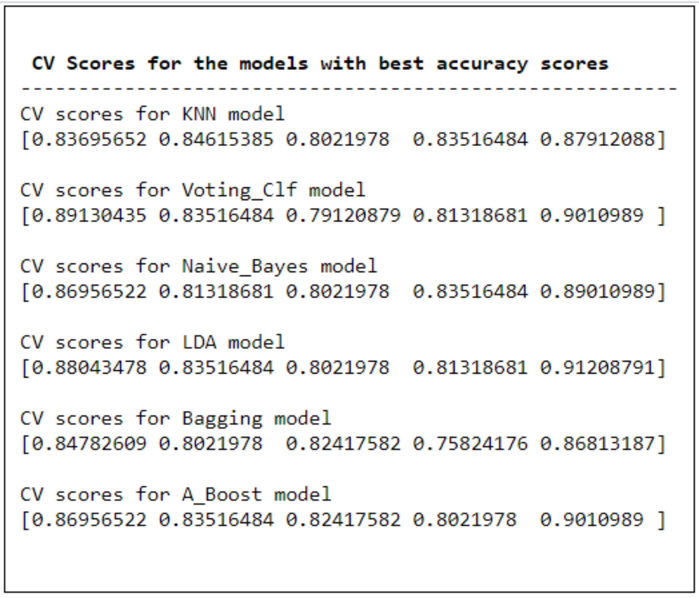
- SMOTE an over sampling technique is used to balance the variables that are originally imbalanced. The given data set is analysed by using SMOTE and results are given below.
- 1. X and y data are exported to Project ML Q1 SMOTE file.
- 2. The data is analysed after implementing SMOTE technique (Over sampling as the target class is imbalanced)
- 3. Naive Bayes, KNN, Logistic regression, LDA and Bagging and Boosting are executed on the new data.
- 4. No accuracy improvements.
- 5. Hence, for further analysis we use the original data without SMOTE.

* P.S Please refer to 'Project ML Q1 SMOTE' codebook

<u>Fig 1.9.5 Model Metrics Dataframe:</u>



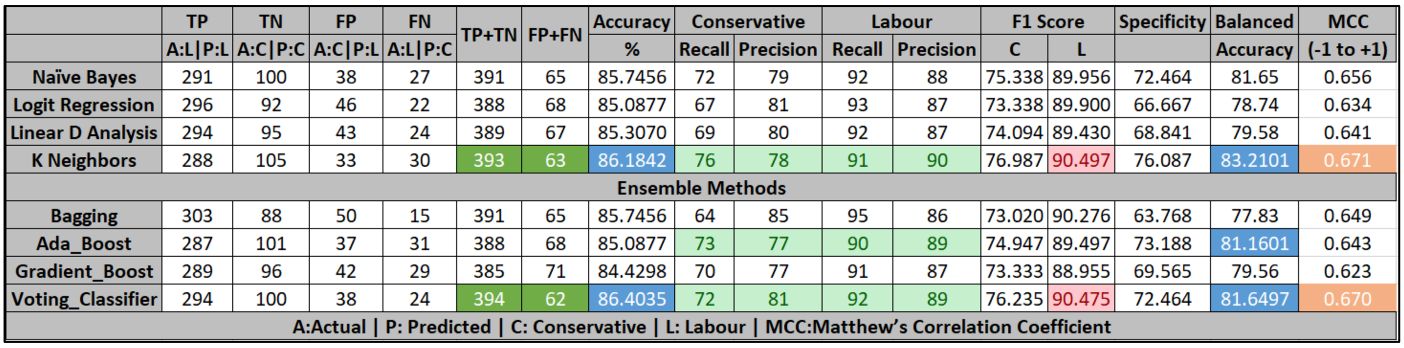
	Accuracy	Roc_Score	F1_Score	Log_Loss
GBoost_Train	0.8982	0.9552	0.9278	3.5158
Voting_clf_Test	0.8640	0.0000	0.9046	4.6961
KNN_Test	0.8618	0.8974	0.9014	4.7719
NB_Test	0.8575	0.9125	0.8995	4.9234
Bag_Test	0.8575	0.9191	0.9031	4.9234
LDA_Test	0.8531	0.9144	0.8977	5.0749
LR_Test	0.8509	0.9109	0.8970	5.1506
ABoost_Test	0.8509	0.9119	0.8941	5.1506
Voting_clf_Train	0.8445	0.0000	0.8904	5.3713
GBoost_Test	0.8443	0.9075	0.8906	5.3778
KNN_Train	0.8398	0.9072	0.8876	5.5341
ABoost_Train	0.8341	0.8869	0.8812	5.7294
Bag_Train	0.8322	0.9012	0.8853	5.7945
LR_Train	0.8303	0.8770	0.8816	5.8596
LDA_Train	0.8228	0.8769	0.8747	6.1201
NB_Train	0.8200	0.8732	0.8709	6.2177



```
 CV Scores for the models with best accuracy scores
-------------------------------------------------------
CV scores for KNN model
[0.83695652 0.84615385 0.8021978  0.83516484 0.87912088]

CV scores for Voting_Clf model
[0.89130435 0.83516484 0.79120879 0.81318681 0.9010989 ]

CV scores for Naive_Bayes model
[0.86956522 0.81318681 0.8021978  0.83516484 0.89010989]

CV scores for LDA model
[0.88043478 0.83516484 0.8021978  0.81318681 0.91208791]

CV scores for Bagging model
[0.84782609 0.8021978  0.82417582 0.75824176 0.86813187]

CV scores for A_Boost model
[0.86956522 0.83516484 0.82417582 0.8021978  0.9010989 ]
```

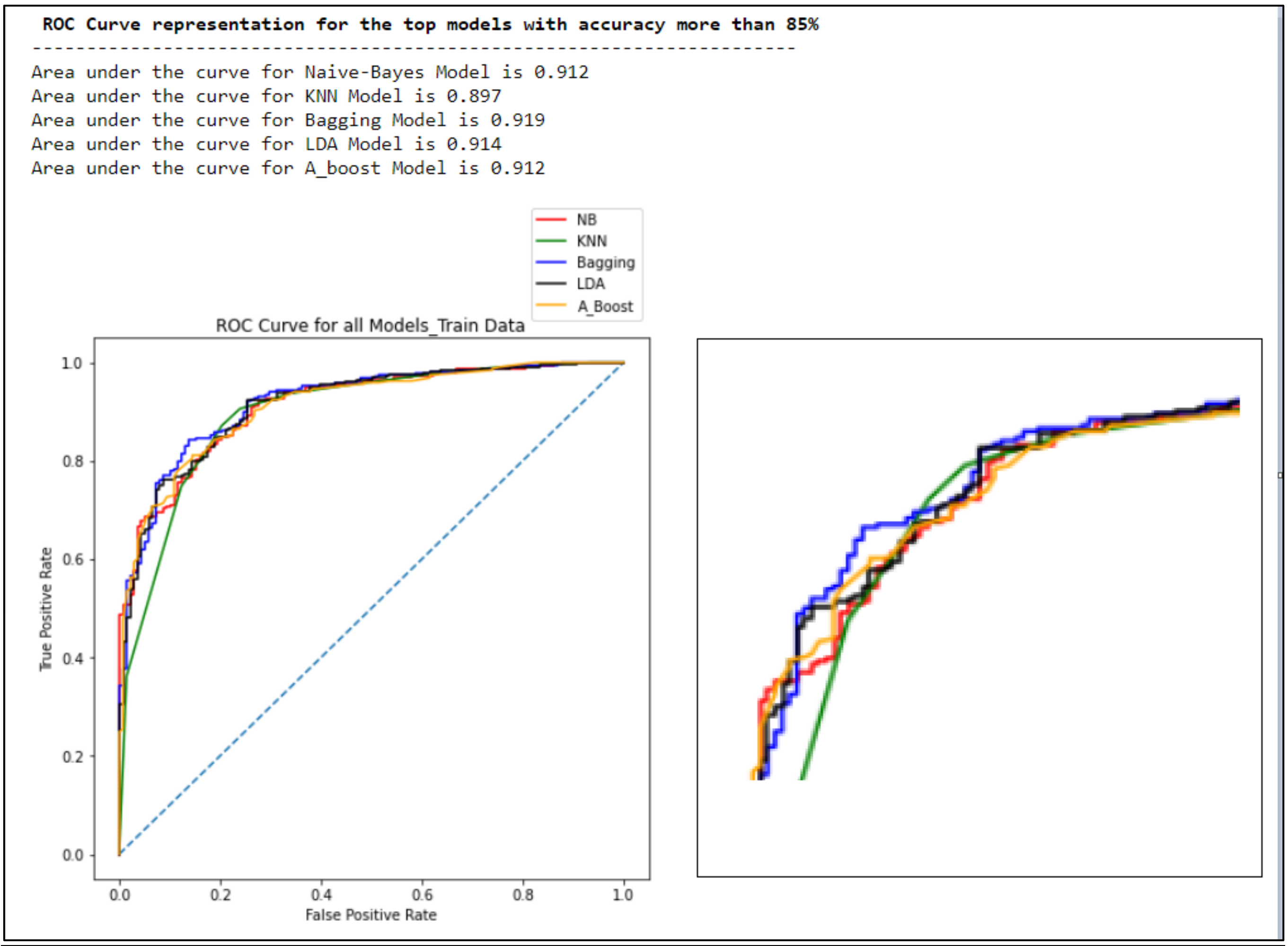
** No Over fitting in any model after tuning the hyper parameters. All the models showed > 80% accuracy on the test data.

Fig 1.9.6: Comparison of Recall, Precision and Confusion Matrix between each model.



	TP	TN	FP	FN	TP+TN	FP+FN	Accuracy	Conservative		Labour		F1 Score		Specificity	Balanced	MCC
	A:L|P:L	A:C|P:C	A:C|P:L	A:L|P:C			%	Recall	Precision	Recall	Precision	C	L		Accuracy	(-1 to +1)
Naïve Bayes	291	100	38	27	391	65	85.7456	72	79	92	88	75.338	89.956	72.464	81.65	0.656
Logit Regression	296	92	46	22	388	68	85.0877	67	81	93	87	73.338	89.900	66.667	78.74	0.634
Linear D Analysis	294	95	43	24	389	67	85.3070	69	80	92	87	74.094	89.430	68.841	79.58	0.641
K Neighbors	288	105	33	30	393	63	86.1842	76	78	91	90	76.987	90.497	76.087	83.2101	0.671
Ensemble Methods																
Bagging	303	88	50	15	391	65	85.7456	64	85	95	86	73.020	90.276	63.768	77.83	0.649
Ada_Boost	287	101	37	31	388	68	85.0877	73	77	90	89	74.947	89.497	73.188	81.1601	0.643
Gradient_Boost	289	96	42	29	385	71	84.4298	70	77	91	87	73.333	88.955	69.565	79.56	0.623
Voting_Classifier	294	100	38	24	394	62	86.4035	72	81	92	89	76.235	90.475	72.464	81.6497	0.670
A:Actual | P: Predicted | C: Conservative | L: Labour | MCC:Matthew's Correlation Coefficient																

Fig 1.9.7: AUC ROC Curve for Models:



```
ROC Curve representation for the top models with accuracy more than 85%
----------------------------------------------------------------------
Area under the curve for Naive-Bayes Model is 0.912
Area under the curve for KNN Model is 0.897
Area under the curve for Bagging Model is 0.919
Area under the curve for LDA Model is 0.914
Area under the curve for A_boost Model is 0.912
```

Recall: True Positive/True Positive + False Negative

Precision: True Positive / True Positive + False Positive

Specificity: True Negative / True Negative + False Positive

Balanced Accuracy: It is the average between Sensitivity and Specificity. As the given data has imbalance in the target variable, we can rely on the balanced accuracy to select the optimized model.

The Matthews correlation coefficient (MCC): It is least influenced by imbalanced data. It is a correlation coefficient between the observed and predicted classifications

1. *K nearest model has good accuracy score, high F1 score and low log loss. The recall and the precision for Class 0 and 1 are equal. The Balanced accuracy and the MCC is high among all the models. When the difference between model accuracy and balanced accuracy is low, it means the model has performed well despite the imbalance in target class. But the AUC ROC score is less for KNN.*
2. *The next best performing individual model is Naïve bayes model. The given data has low multi-collinearity, hence NB has performed well. It has good accuracy and F1 score. The recall and precision are equal for both target classes. This has good AUC ROC score.*
3. Logistic regression and Linear discriminant analysis though it had good accuracy scores the log loss values are slightly high. The difference between the Model accuracy and the balanced accuracy are high. *Both the classifiers have taken the advantage of the majority class.*
4. The ensemble algorithms Voting classifier have high model accuracy, low log loss, high F1 score and equal recall and precision values for both the classes. The MCC is high among all the ensemble models. The balanced accuracy is not very low from the model accuracy. The MCC is high among all other ensemble models.
5. The next best performing ensemble model is Ada boosting algorithm. The only drawback is, it captured less True predictions and high False predictions.
6. The gradient boost has high accuracy score for training but it shows very less accuracy score in test. The total true predictions are lowest and the total False predictions are highest in this model. The recall and precision between both the classes.
7. The ensemble bagging model though it has captured high true positives and has high area under curve score. But the difference between the model and balanced accuracy is very high. This clearly shows that Bagging model has taken the advantage of imbalance in the target class.
8. No large variations in CV scores are found
9. *For the given problem statement, Recall and Precision both are equally important. A voter actually voted for Labour party predicted as conservative and vice versa, both will impact the exit poll which will impact the overall win and seats covered.*

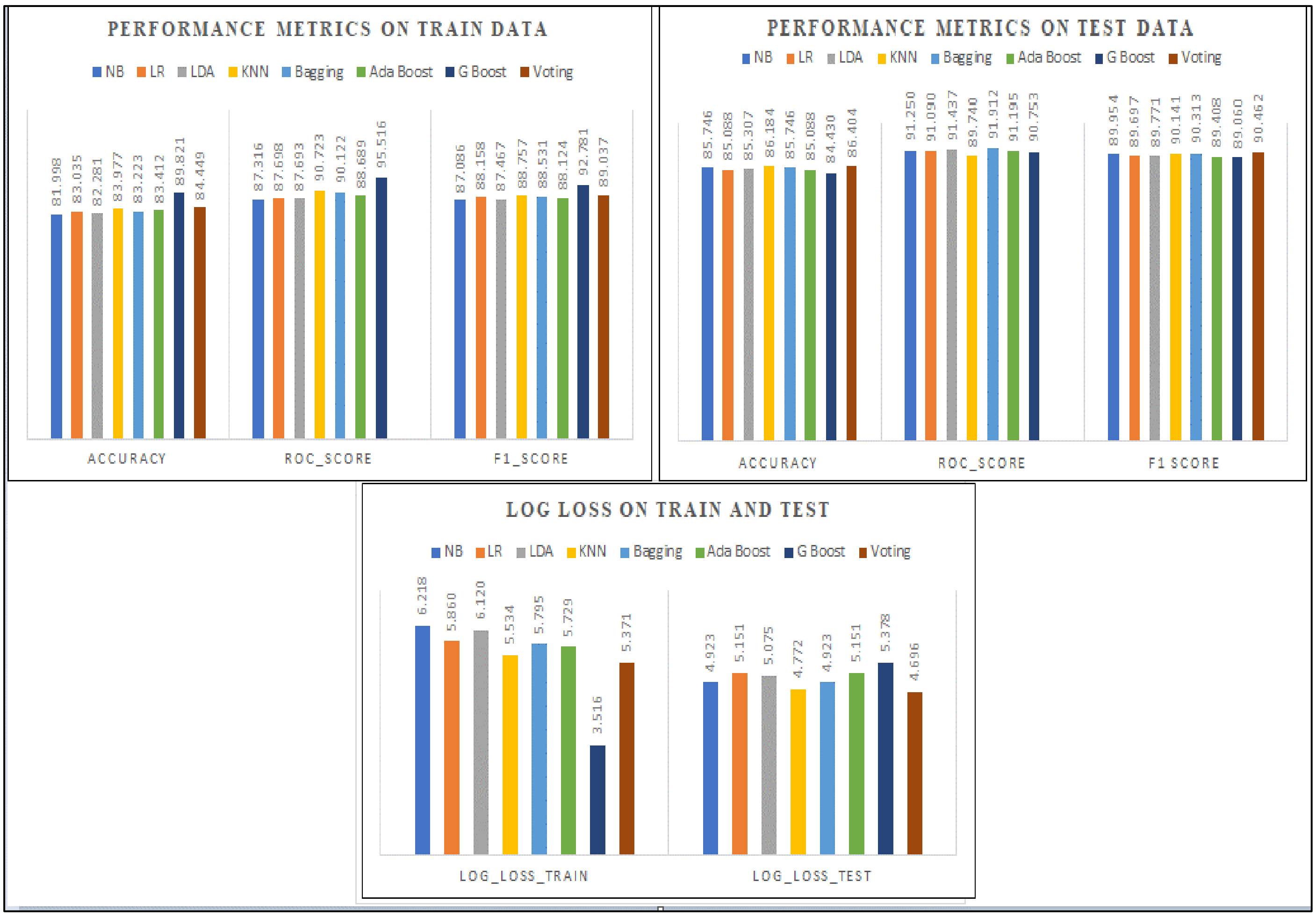
10. *Hence, a model which gives almost equal score for both recall and precision should be considered.*

Best Model – Individual: K nearest neighbour and Naïve Bayes Models performed well.

Best Model – Ensemble: Voting Classifier and Ada boosting

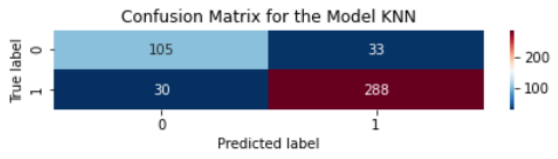
Instead of taking the prediction results only from KNN model, we can take the predictions of the Voting Classifier. The ensemble method Voting Classifier shows highest accuracy and very low log loss. Collection of different models helps to get 'Bias-Variance trade-off'. The mechanism for improved performance with ensembles is often the reduction in the variance component of prediction errors made by the contributing models. It helps to reduce the variance and improves accuracies. *Combining several weak models can result in what we call a strong learner. Though we don't have any weak models, we have improved the accuracy score and reduced the log loss using ensemble technique.*

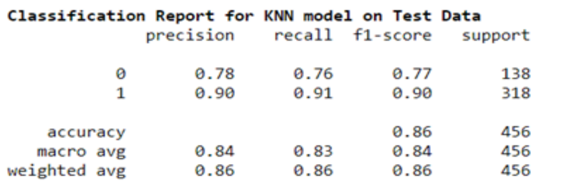
<u>Fig 1.9.8: Graphical representation of Performance metrics of all models:</u>



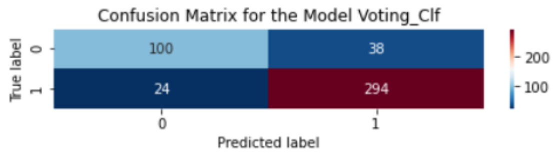
- All the models except Gradient Boosting performed equally or higher than the train data. Only Gradient boost model has shown good scores in train and low scores in test data.
- The log loss in train for Gradient boosting is 3.516 but in Test it is 5.378.

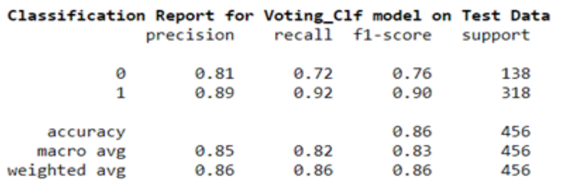
<u>Fig1.9.9: Confusion Matrix and Classification report for all the models:</u>



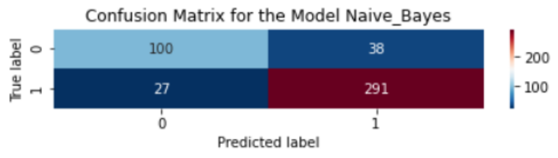


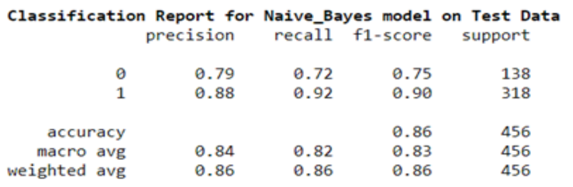
```
Classification Report for KNN model on Test Data
              precision    recall  f1-score   support

           0       0.78      0.76      0.77       138
           1       0.90      0.91      0.90       318

    accuracy                           0.86       456
   macro avg       0.84      0.83      0.84       456
weighted avg       0.86      0.86      0.86       456

---------------------------------------------------------
```



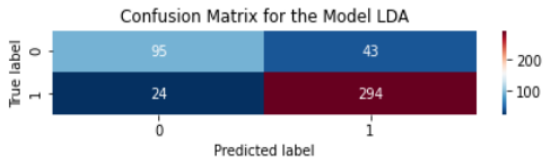


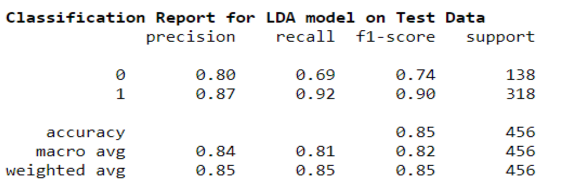
```
Classification Report for Voting_Clf model on Test Data
              precision    recall  f1-score   support

           0       0.81      0.72      0.76       138
           1       0.89      0.92      0.90       318

    accuracy                           0.86       456
   macro avg       0.85      0.82      0.83       456
weighted avg       0.86      0.86      0.86       456

---------------------------------------------------------
```



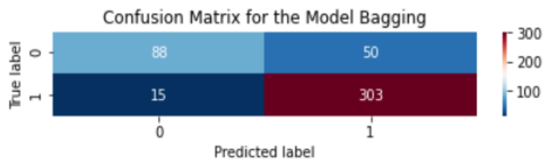


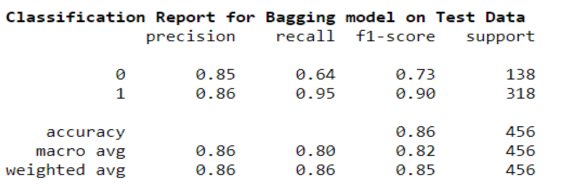
```
Classification Report for Naive_Bayes model on Test Data
              precision    recall  f1-score   support

           0       0.79      0.72      0.75       138
           1       0.88      0.92      0.90       318

    accuracy                           0.86       456
   macro avg       0.84      0.82      0.83       456
weighted avg       0.86      0.86      0.86       456

---------------------------------------------------------
```



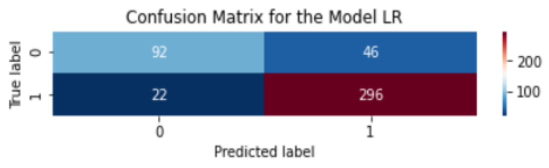


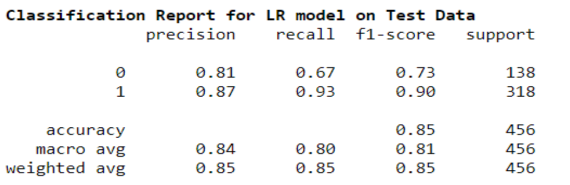
```
Classification Report for LDA model on Test Data
              precision    recall  f1-score   support

           0       0.80      0.69      0.74       138
           1       0.87      0.92      0.90       318

    accuracy                           0.85       456
   macro avg       0.84      0.81      0.82       456
weighted avg       0.85      0.85      0.85       456

---------------------------------------------------------
```



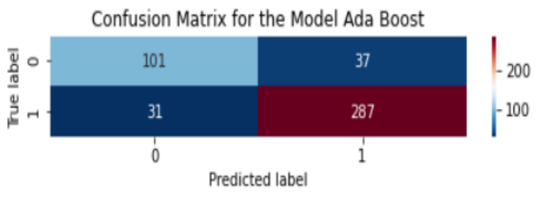


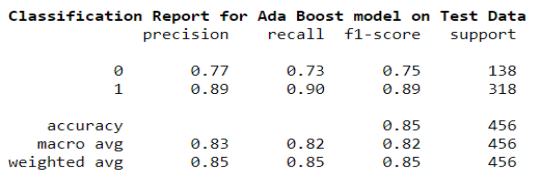
```
Classification Report for Bagging model on Test Data
              precision    recall  f1-score   support

           0       0.85      0.64      0.73       138
           1       0.86      0.95      0.90       318

    accuracy                           0.86       456
   macro avg       0.86      0.80      0.82       456
weighted avg       0.86      0.86      0.85       456

---------------------------------------------------------
```



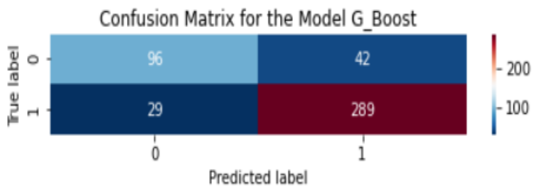


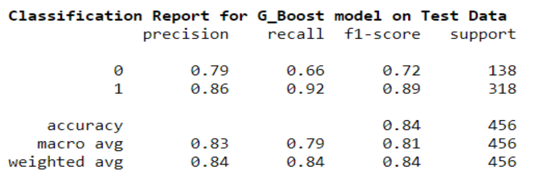
```
Classification Report for LR model on Test Data
              precision    recall  f1-score   support

           0       0.81      0.67      0.73       138
           1       0.87      0.93      0.90       318

    accuracy                           0.85       456
   macro avg       0.84      0.80      0.81       456
weighted avg       0.85      0.85      0.85       456

---------------------------------------------------------
```





```
Classification Report for Ada Boost model on Test Data
              precision    recall  f1-score   support

           0       0.77      0.73      0.75       138
           1       0.89      0.90      0.89       318

    accuracy                           0.85       456
   macro avg       0.83      0.82      0.82       456
weighted avg       0.85      0.85      0.85       456

---------------------------------------------------------
```





```
Classification Report for G_Boost model on Test Data
              precision    recall  f1-score   support

           0       0.79      0.66      0.72       138
           1       0.86      0.92      0.89       318

    accuracy                           0.84       456
   macro avg       0.83      0.79      0.81       456
weighted avg       0.84      0.84      0.84       456

---------------------------------------------------------
```

1.8 Based on these predictions, what are the insights? (5 marks)

Fig 2.1: Plots showing the Important features for the predictions and its coefficients:

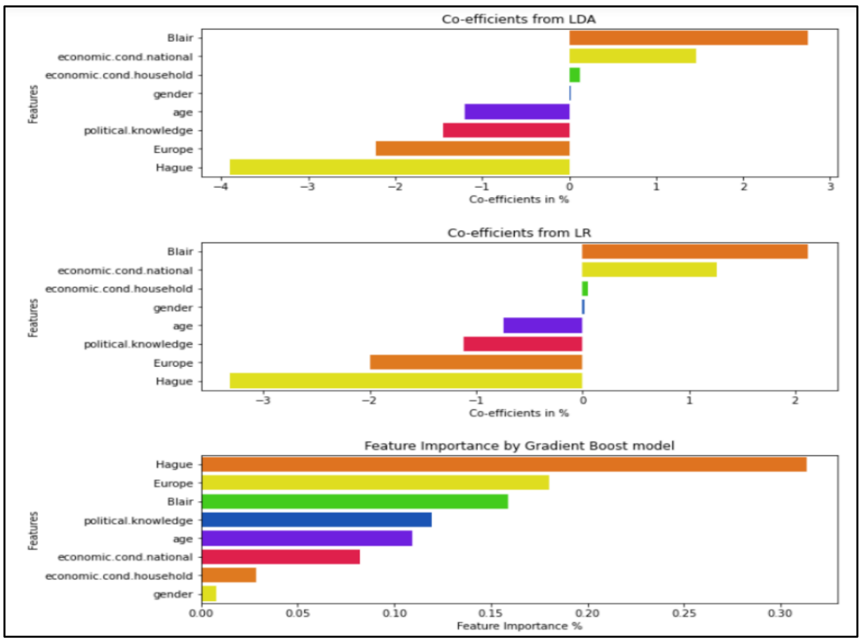
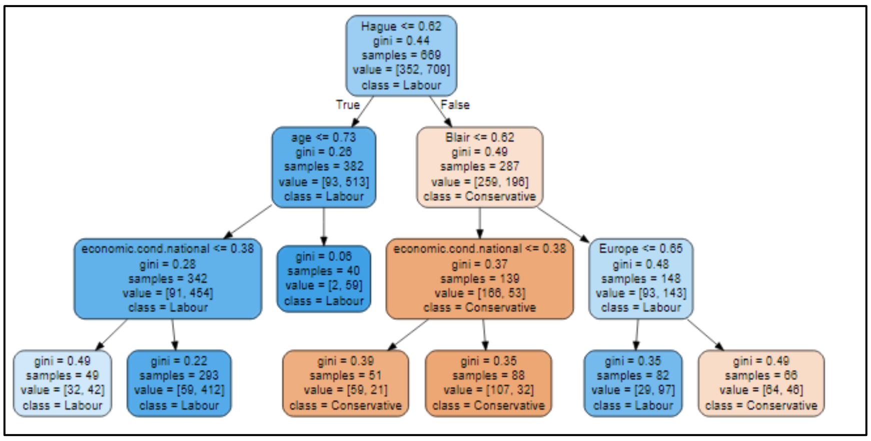
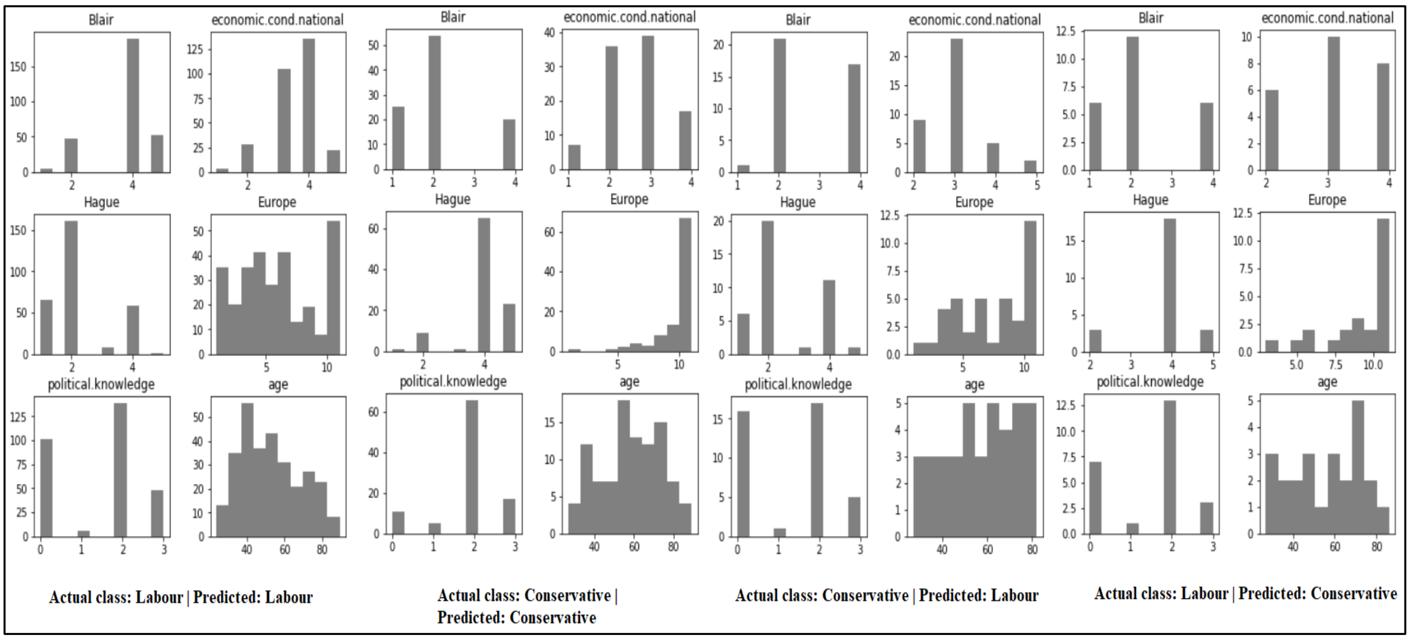


Fig 2.2: A Random tree from the Random Forest



1. Fig 2.1, shows the graphical representation of the coefficients for all the features from the model Logistic regression and Linear discriminant analysis. By looking at the plot we can understand that William Hague, Tony Blair, Europe, Political knowledge and national economic conditions are the important predictors.
2. The Feature importance from the Gradient boost model also shows the similar features as important predictors. Gender and household economic conditions are the weak predictors of the vote.
3. Fig 2.2, shows a graphical representation of a random tree from the random forest. The root node is the feature ‘Hague’ with lowest gini index. If voters have given good ratings to Hague, the next split is using the ‘age’ feature. If the ratings are very low, the other side of the split is using the ‘Blair’ feature.

Fig: 2.3: Plots for True Positive, True Negative, False positive and False Negative



Insights:

1. A voter with high ratings for Tony Blair and low ratings for William Hague votes for Labour party.
2. A voter with high ratings for William Hague and low ratings for Tony Blair votes for Conservative party.
3. Voters with Eurosceptic attitude prefer Conservative party. They do not have neutral opinion towards the integration.
4. Voters of Labour party believe in good national economic conditions.
5. The median age of the voters of conservative party are higher than voters of labour party. From this we can infer that, people who are aged prefer conservative party.

Conclusion:

With a data set of 1525 voters and 9 variables, different models like Naïve Bayes, K nearest neighbors, Logistic regression and Linear discriminant analysis are executed. The ensemble methods like Bagging, Boosting and Voting classifiers are used to get Bias-Variance trade off. The models which are over fitted and the models which had low accuracy scores are tuned using the Gird search parameters. The performance of the models improved after applying Grid search. The best or optimized model for this problem statement is K nearest neighbor with n_neighbors =11 and Naïve Bayes model. The predictions from all the models are combined using the Voting Classifier and the best predictions are calculated using 'soft' voting technique. On analysing the predictions and the actual results, high and low ratings for William Hague, Tony Blair, Europe and Political knowledge are the key predictors.

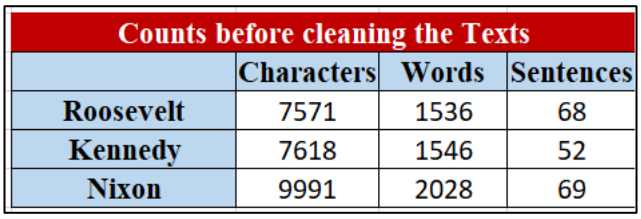
Problem Statement: 2

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

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

2.1 Find the number of characters, words, and sentences for the mentioned documents. – 3 Marks

Table 1.1: Counts before cleaning the text:



Counts before cleaning the Texts			
	Characters	Words	Sentences
Roosevelt	7571	1536	68
Kennedy	7618	1546	52
Nixon	9991	2028	69

The inaugural speech by Nixon, has high number of characters, words and sentences. The above table shows the count of characters, words and sentences for all the three speeches before cleaning the texts.

2.2 Remove all the stopwords from all three speeches. – 3 Marks

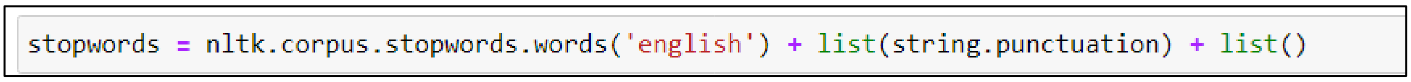
Stopwords: Stop words are the words which are very common in text documents such as a, an, the, you, your, etc. The Stop Words highly appear in text documents. However, they are not being helpful for text analysis in many of the cases, So, it is better to remove from the text.

Punctuations: Punctuations like full-stop, comma, hyphen may appear many times in the text. System captures the punctuations as characters. They are not helpful for the text analysis. So, it is better to remove from the text.

Numbers and Special characters: Numbers and special characters like ‘#’, ‘—‘,’@’ should also be removed from the text.

Steps followed:

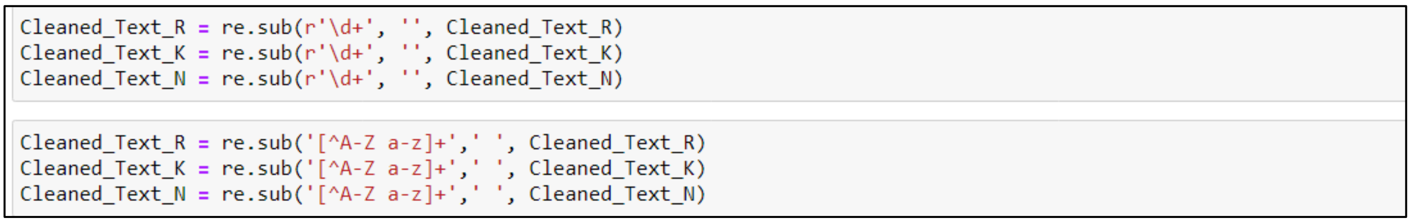
1. All the words are converted into ‘lower case’
2. From nltk.corpus toolkit, the stopwords for language ‘English’ are imported. By using the below code, the stopwords are removed.



```
stopwords = nltk.corpus.stopwords.words('english') + list(string.punctuation) + list()
```

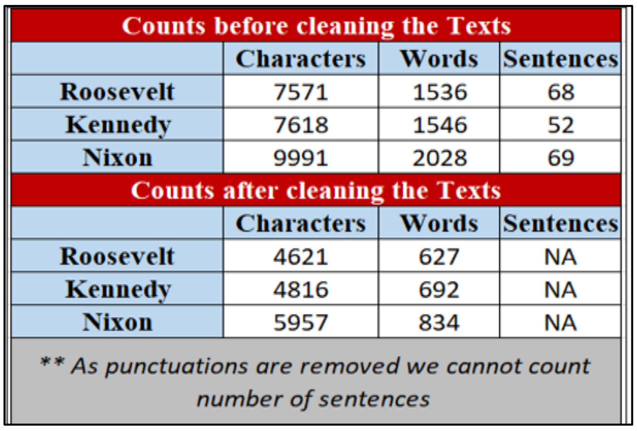
3. Using the below code, numbers and special characters are also removed.

Fig:1.1 Code to remove numbers and Special characters using regular expressions:



```
Cleaned_Text_R = re.sub(r'\d+', '', Cleaned_Text_R)
Cleaned_Text_K = re.sub(r'\d+', '', Cleaned_Text_K)
Cleaned_Text_N = re.sub(r'\d+', '', Cleaned_Text_N)

Cleaned_Text_R = re.sub('[^A-Z a-z]+',' ', Cleaned_Text_R)
Cleaned_Text_K = re.sub('[^A-Z a-z]+',' ', Cleaned_Text_K)
Cleaned_Text_N = re.sub('[^A-Z a-z]+',' ', Cleaned_Text_N)
```

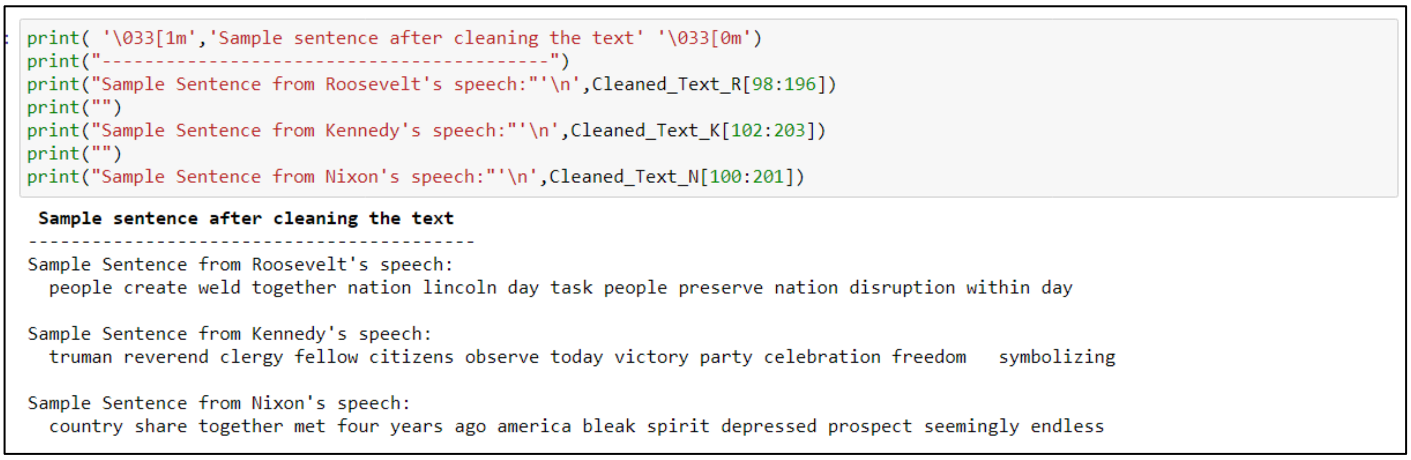
Table 1.2: Counts before and after cleaning the text:



Counts before cleaning the Texts			
	Characters	Words	Sentences
Roosevelt	7571	1536	68
Kennedy	7618	1546	52
Nixon	9991	2028	69
Counts after cleaning the Texts			
	Characters	Words	Sentences
Roosevelt	4621	627	NA
Kennedy	4816	692	NA
Nixon	5957	834	NA

*** As punctuations are removed we cannot count number of sentences*

Fig:1.2 Sample Sentences from all three speeches after cleaning:



```
print( '\033[1m','Sample sentence after cleaning the text' '\033[0m')
print("----------------------------------------")
print("Sample Sentence from Roosevelt's speech:"'\n',Cleaned_Text_R[98:196])
print("")
print("Sample Sentence from Kennedy's speech:"'\n',Cleaned_Text_K[102:203])
print("")
print("Sample Sentence from Nixon's speech:"'\n',Cleaned_Text_N[100:201])
```

```
 Sample sentence after cleaning the text
----------------------------------------
Sample Sentence from Roosevelt's speech:
  people create weld together nation lincoln day task people preserve nation disruption within day

Sample Sentence from Kennedy's speech:
  truman reverend clergy fellow citizens observe today victory party celebration freedom   symbolizing

Sample Sentence from Nixon's speech:
  country share together met four years ago america bleak spirit depressed prospect seemingly endless
```

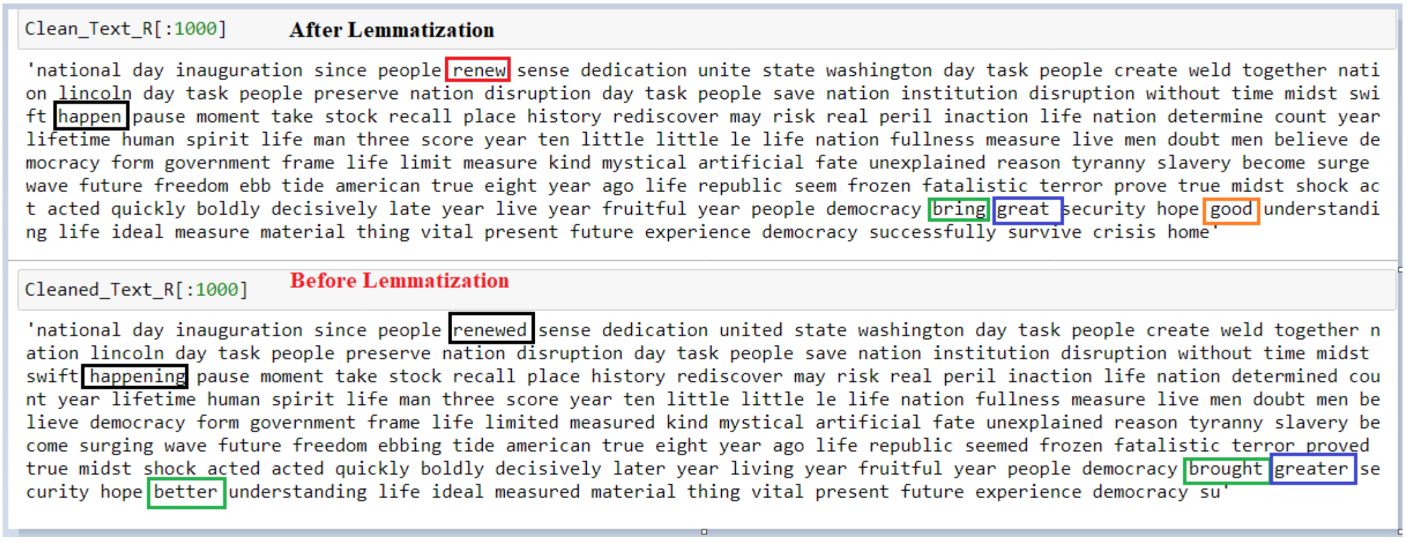
- The sample sentence does not have any punctuations, number and stop words.

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)

Before analysing the words that occur more frequently, we need to clean the words that are inside each text using techniques like stemming or lemmatization.

Stemming and lemmatization are methods used by search engines and chatbots to analyze the meaning behind a word. Stemming uses the stem of the word, while lemmatization uses the context in which the word is being used. For the given text, Lemmatization is used for text normalization.

Fig: 1.3 Difference in words from Roosevelt's speech before and after lemmatization:

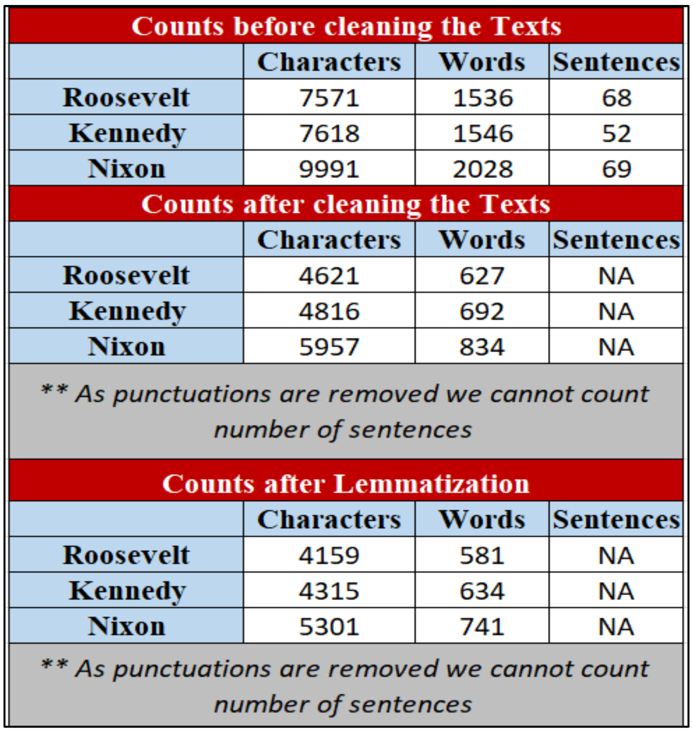


```
Clean_Text_R[:1000]        After Lemmatization

'national day inauguration since people renew sense dedication unite state washington day task people create weld together nati
on lincoln day task people preserve nation disruption day task people save nation institution disruption without time midst swi
ft happen pause moment take stock recall place history rediscover may risk real peril inaction life nation determine count year
lifetime human spirit life man three score year ten little little le life nation fullness measure live men doubt men believe de
mocracy form government frame life limit measure kind mystical artificial fate unexplained reason tyranny slavery become surge
wave future freedom ebb tide american true eight year ago life republic seem frozen fatalistic terror prove true midst shock ac
t acted quickly boldly decisively late year live year fruitful year people democracy bring great security hope good understandi
ng life ideal measure material thing vital present future experience democracy successfully survive crisis home'
```

```
Cleaned_Text_R[:1000]        Before Lemmatization

'national day inauguration since people renewed sense dedication united state washington day task people create weld together n
ation lincoln day task people preserve nation disruption day task people save nation institution disruption without time midst
swift happening pause moment take stock recall place history rediscover may risk real peril inaction life nation determined cou
nt year lifetime human spirit life man three score year ten little little le life nation fullness measure live men doubt men be
lieve democracy form government frame life limited measured kind mystical artificial fate unexplained reason tyranny slavery be
come surging wave future freedom ebbing tide american true eight year ago life republic seemed frozen fatalistic terror proved
true midst shock acted acted quickly boldly decisively later year living year fruitful year people democracy brought greater se
curity hope better understanding life ideal measured material thing vital present future experience democracy su'
```

- The above picture contains 1000 cleaned words from Roosevelt's speech. The words are compared before and after lemmatization. Some of the examples are high lightened.
- Renewed → renew, brought→ bring, better→good.

Table 1.3: Counts after lemmatization:

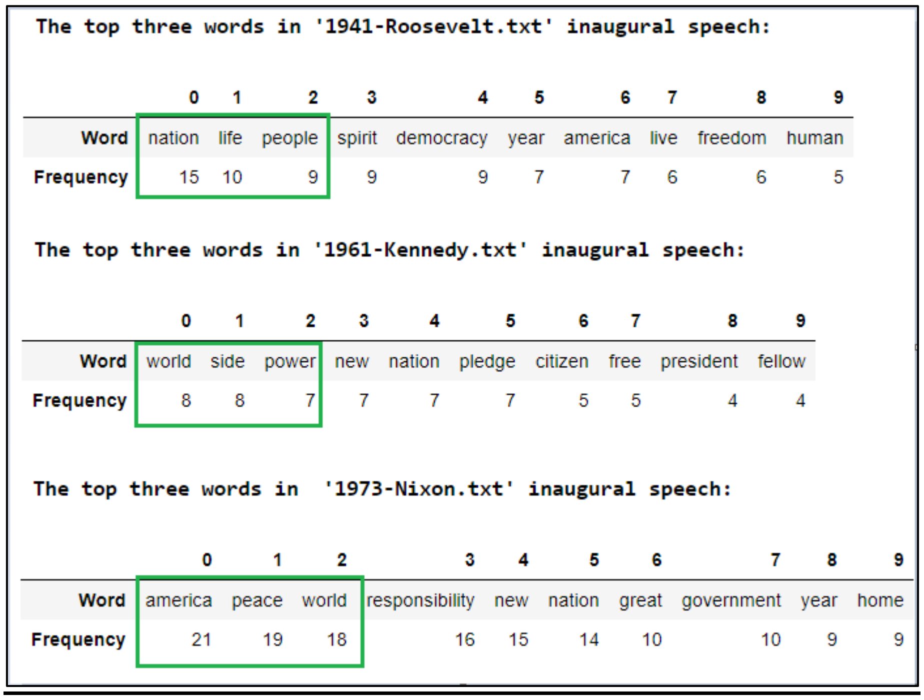


Counts before cleaning the Texts			
	Characters	Words	Sentences
Roosevelt	7571	1536	68
Kennedy	7618	1546	52
Nixon	9991	2028	69

Counts after cleaning the Texts			
	Characters	Words	Sentences
Roosevelt	4621	627	NA
Kennedy	4816	692	NA
Nixon	5957	834	NA
*** As punctuations are removed we cannot count number of sentences*			

Counts after Lemmatization			
	Characters	Words	Sentences
Roosevelt	4159	581	NA
Kennedy	4315	634	NA
Nixon	5301	741	NA
*** As punctuations are removed we cannot count number of sentences*			

Fig:1.4 The most common word and the top three words from each speech:



The top three words in '1941-Roosevelt.txt' inaugural speech:

	0	1	2	3	4	5	6	7	8	9
Word	nation	life	people	spirit	democracy	year	america	live	freedom	human
Frequency	15	10	9	9	9	7	7	6	6	5

The top three words in '1961-Kennedy.txt' inaugural speech:

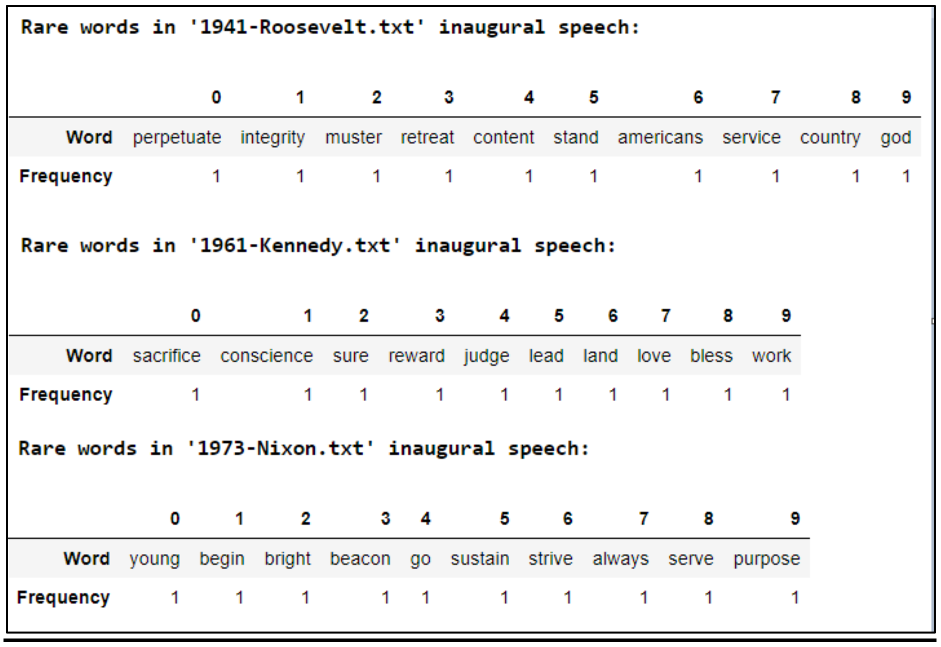
	0	1	2	3	4	5	6	7	8	9
Word	world	side	power	new	nation	pledge	citizen	free	president	fellow
Frequency	8	8	7	7	7	7	5	5	4	4

The top three words in '1973-Nixon.txt' inaugural speech:

	0	1	2	3	4	5	6	7	8	9
Word	america	peace	world	responsibility	new	nation	great	government	year	home
Frequency	21	19	18	16	15	14	10	10	9	9

- We can add 'side' as a stop word and check the other important words.
- The words 'nation','world','america' are the three top words from Roosevelt, Kennedy and Nixon Speech. The top three words are high lighted using green box.

Fig:1.5 The most rarely used words in each speech:



Rare words in '1941-Roosevelt.txt' inaugural speech:

	0	1	2	3	4	5	6	7	8	9
Word	perpetuate	integrity	muster	retreat	content	stand	americans	service	country	god
Frequency	1	1	1	1	1	1	1	1	1	1

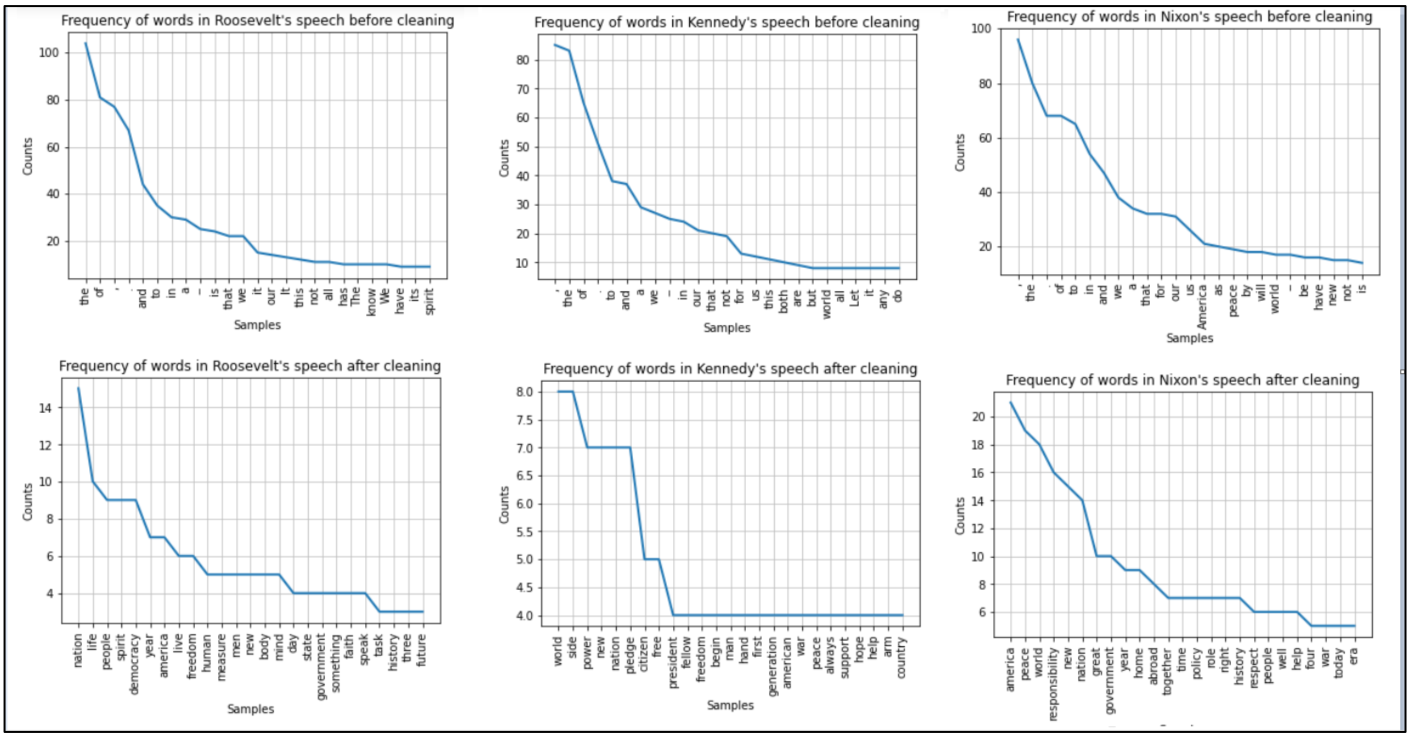
Rare words in '1961-Kennedy.txt' inaugural speech:

	0	1	2	3	4	5	6	7	8	9
Word	sacrifice	conscience	sure	reward	judge	lead	land	love	bless	work
Frequency	1	1	1	1	1	1	1	1	1	1

Rare words in '1973-Nixon.txt' inaugural speech:

	0	1	2	3	4	5	6	7	8	9
Word	young	begin	bright	beacon	go	sustain	strive	always	serve	purpose
Frequency	1	1	1	1	1	1	1	1	1	1

<u>Fig 1.6: Word Frequency distribution plots for all three inaugural speeches before and after cleaning:</u>

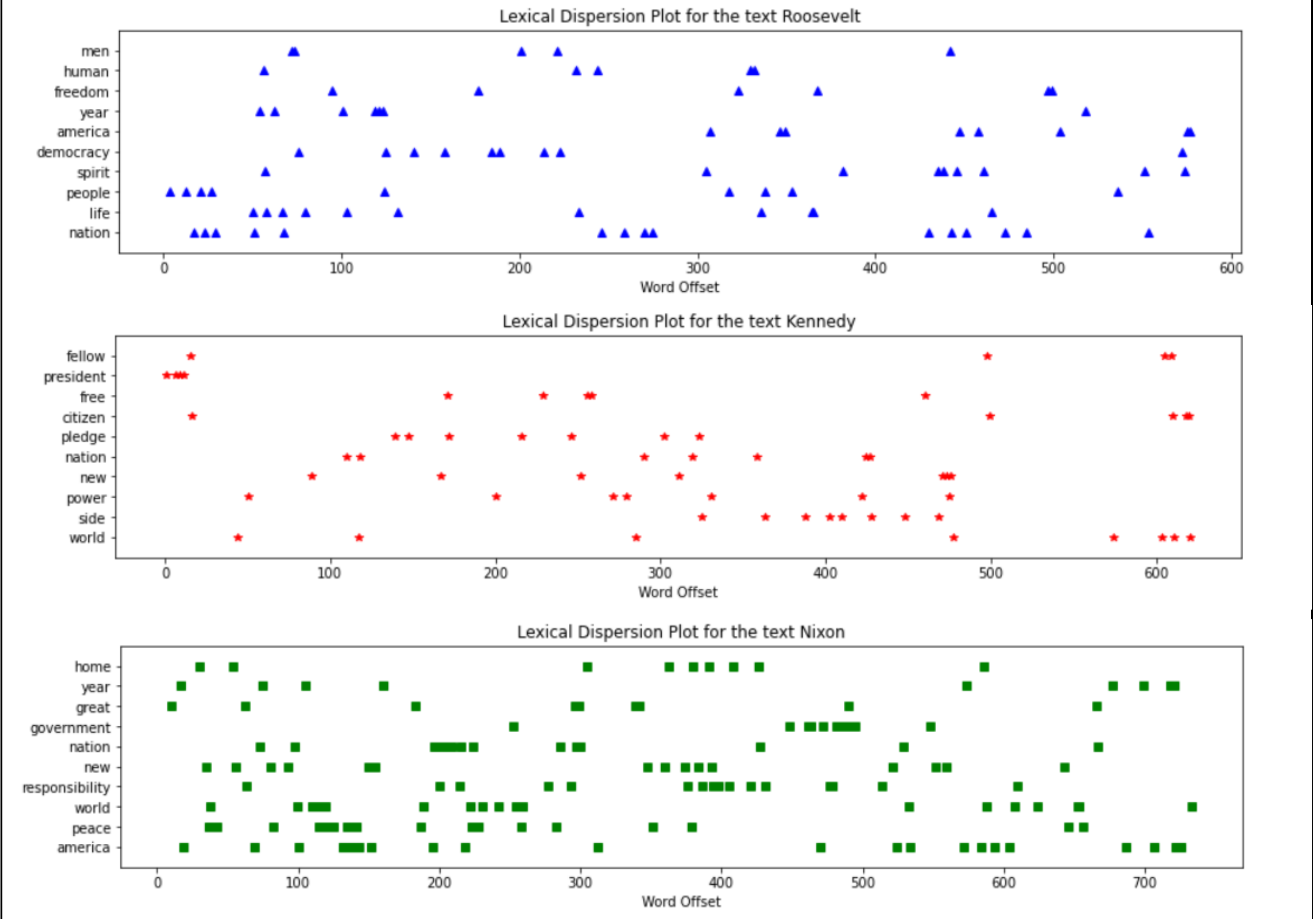


- The frequency for stop words like ‘the, of, in, and, to, etc are high in the plots before cleaning.
- In the second plot i.e., after cleaning the text, the words are entirely different from the first plot. Indeed, these words are useful and important for text analysis.

Lexical Dispersion plot:

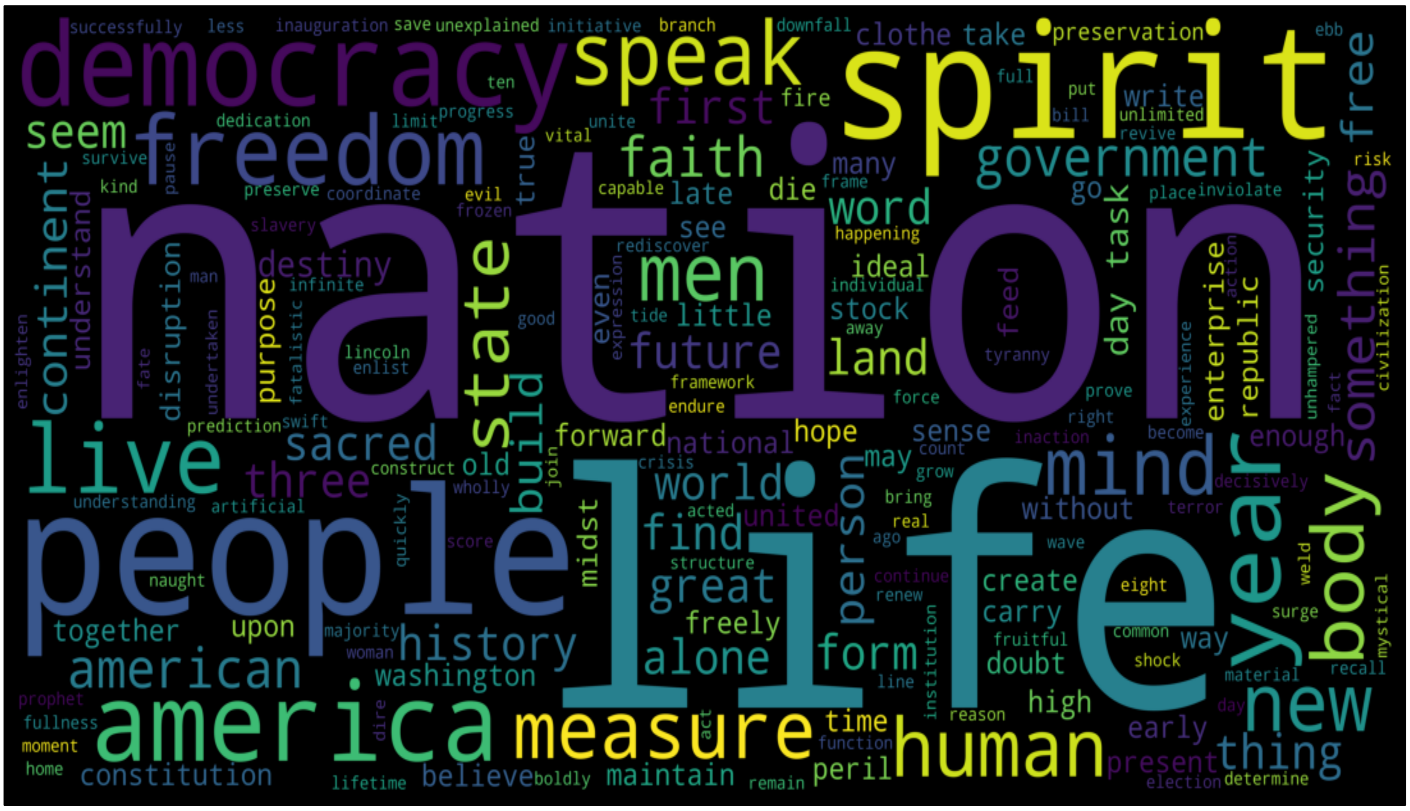
Lexical dispersion is a measure of a word's homogeneity across the parts of a corpus. Lexical dispersion illustrates the homogeneity of a word (or set of words) across the documents of a corpus.

Fig:1.7 Lexical Dispersion plot



1. Lexical dispersion plot-Roosevelt: The words like 'nation', 'life', 'people', 'democracy' are used by Roosevelt throughout the speech. But words like 'men' and 'america' is used at the beginning and ending respectively.
2. Lexical dispersion plot-Kennedy: The words like 'side, 'power', 'new', 'nation' are used by Kennedy throughout the speech. But words like 'president and 'citizen are used only at the beginning and ending respectively.
3. Lexical dispersion plot-Nixon: The frequency of the words in Nixon text is very different from the other two. All the most common words are spread across the speech.

Word Cloud for Roosevelt Text:



Word Cloud for Kennedy Text:

world
side
new
power
nation
citizen
first
free
pledge
fellow
freedom
always
president
american
begin
generation
man
friend
peace
burden
help
dare
country
hand
state
control
support
struggle
unite
call
life
war
hope
right
good
arm
poverty
human
today
final
globe
well
back
word
form
instrument
alliance
never
loyalty
vice
slow
price
eisenhower
finish
observe
host
different
terror
revolution
whose
foe
solemn
faithful
year
celebration
alike
oppose
anew
enemy
venture
long
proud
fight
three
victory
mortal
witness
people
temper
discipline
course
ancient
unwilling
science
torch
summon
renewal
ago
party
problem
instead
bear
may
day
revolutionary
generosity
merely
cultural
go
forth
though
welcome
together
offer
history
place
yet
born
time
negotiate
chief
far
origin
heir
abolish
forget
deed
finally
quarter
hardship
weakness
find
survival
liberty
bring
whether
beyond
join
old
spiritual
nixon
around
fear
pay
swear
belief
commit
reverend
nearly
best
oath
hold
weak
permit
truman
meet
light
seek
become
assure
prescribe
almighty
hard
truly
pass
south
signify
home
undo
change
johnson
absolute
tyranny
mankind
destruction
balance
divide
doubt
powerful
explore
god
challenge
cooperative
ill
heritage
issue
bitter
success
speaker
justice
clergy
much
earth
odds
america
share
century
united
endeavor
little
effort
end
forebear
remember
ally
disease
wish
symbolize
order

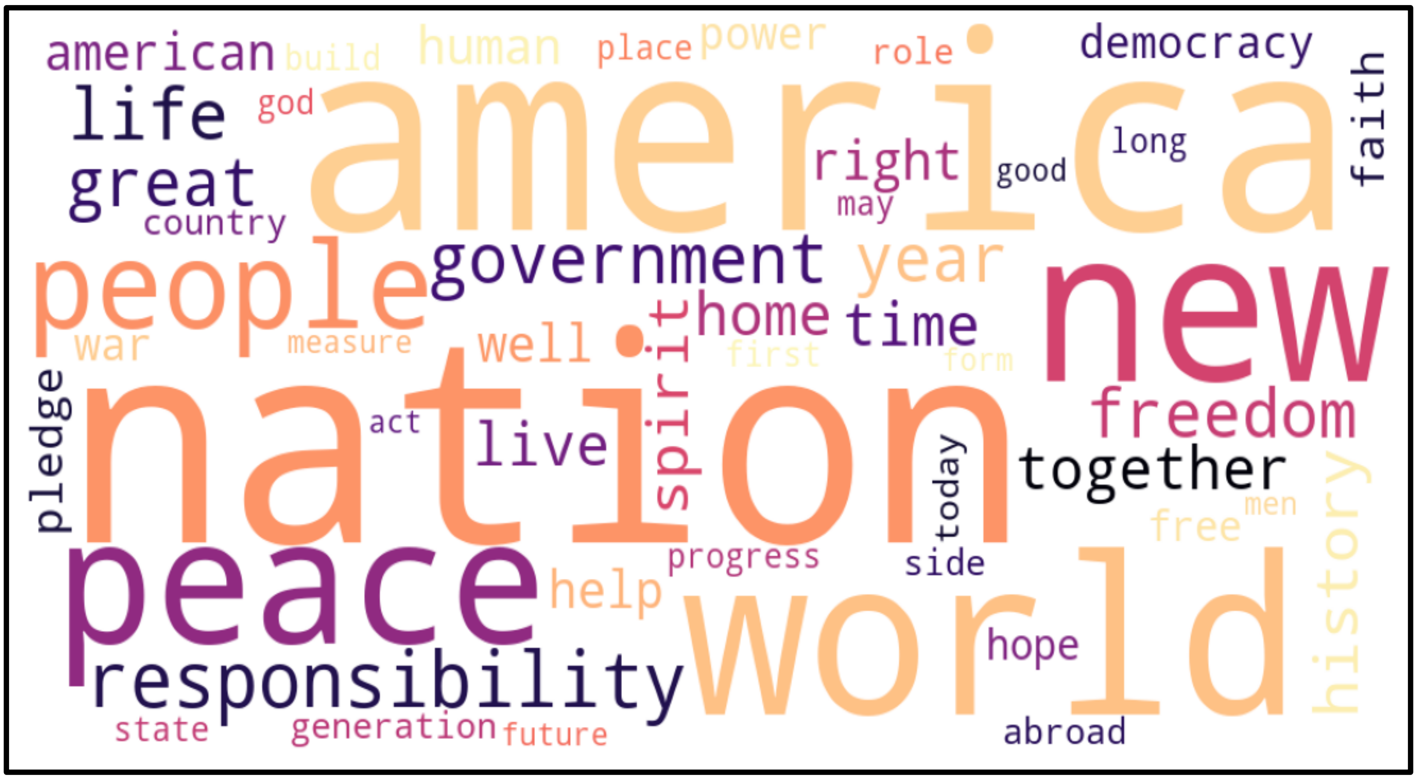
Word Cloud for Nixon Text:

- The word cloud for Nixon text is created using a ‘Cloud Mask’.

progress
conflict
old policy
world
meet
face
destructive
remain
past
reduce
policy
arm
individual
play
together
help
era
greatly
record
last
preserve
challenge
forward
understand
faith
respect
freedom
history
nation
today
role
act
way
expect
opportunity
another
build
retreat
commitment
new
america
place
continue
force
people
great
proud
home
well
right
peace
abroad
live
share
nature
country
others
structure
work
lead
reach
four year
god
ashamed
bold
stand
generation
one
time
responsibility
war
spirit
future
think
long
government
year
century
gladly
need
end
turn away
promise
system
resolve
indispensable
friendship
learn
far
remember
result
bear
power
danger
chance
initiative
important
recognize
renew
dream
toward
require
boldly
unless
shift
good
washington
limitation
merely
confident
strong
traditional
defend
central

Word Cloud for all the three texts / speeches:

1. Word cloud for all the three speeches is created. The below word cloud is the representation of 50 words chosen randomly from the corpus.
2. It contains words *like "people, freedom, nation, peace, faith, responsibility, spirit, hope, great, power, progress, great"*. These words convey that the speech sentiment of all the three leaders were ***'Positive and assuring'.***



Conclusion:

The inaugural speeches of Mr. Roosevelt, Mr. Kennedy and Mr. Nixon are imported from nltk corpus. The characters, words and sentences count before and after cleaning are showed. The cleaning is performed in the below order.

1. Converted into lower case
2. Removed first set of stop words.
3. Removed Punctuations
4. Removed numbers and special characters
5. Removed second set of use less stop words
6. Text normalization using Lemmatization technique.

With the cleaned set of words, Word cloud for three different speeches is created. A final word cloud with maximum of 50 words from all three speeches together are created.

List of libraries imported in the Jupyter Codebook:

Common Libraries:

```
import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

sns.set()

%matplotlib inline

from sklearn.model_selection import train_test_split

from statsmodels.stats.outliers_influence import variance_inflation_factor
```

Problem Statement:1

```
from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

from statsmodels.stats.outliers_influence import variance_inflation_factor

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import MinMaxScaler

from sklearn.naive_bayes import GaussianNB

from sklearn.linear_model import LogisticRegression

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

from sklearn.neighbors import KNeighborsClassifier

from sklearn.ensemble import BaggingClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import GridSearchCV

from sklearn.ensemble import GradientBoostingClassifier

from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import VotingClassifier

from sklearn.metrics import
accuracy_score,roc_auc_score,roc_curve,classification_report,confusion_matrix,log_loss,f1_score,balanced
_accuracy_score

from sklearn.model_selection import cross_val_score,KFold
```

<u>Problem Statement: 2</u>

```
import nltk

import re

import string

nltk.download('inaugural')

from nltk.corpus import inaugural

from nltk.stem import WordNetLemmatizer

from nltk.probability import FreqDist

from wordcloud import WordCloud

from PIL import Image
```