MACHINE LEARNING PROJECT

BUSINESS REPORT

Problem 1:

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

Dataset for Problem: [Election\_Data.xlsx](https://olympus.mygreatlearning.com/courses/78185/files/6830414/download?verifier=W76bmNhSAuCO0CagANt2wUteHz8ui8COIjvMhmuk&wrap=1)

**Description of dataset:**

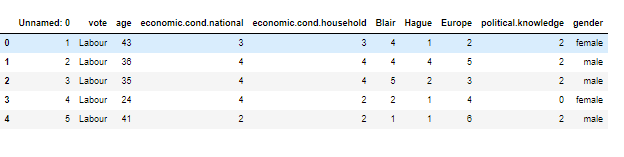
**There is total 9 variables for which data has been collected from 1525 people.They have voted for labour party or conservative party.**

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.**
   1. **Read the dataset. Describe the data briefly. Interpret the inferences for each. Initial steps like head() .info(), Data Types, etc . Null value check, Summary stats, Skewness must be discussed.**

**Load the data:**

el\_df = pd.read\_excel('Election\_Data.xlsx' ,sheet\_name='Election\_Dataset\_Two Classes')

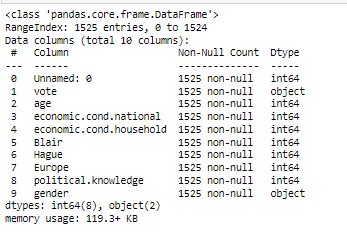
**Head:**

****

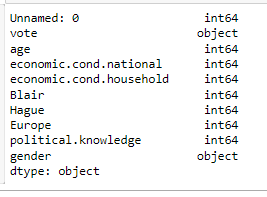
**Shape:**

(1525, 10)

**Data info:**

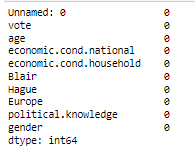
****

**Data types:**

****

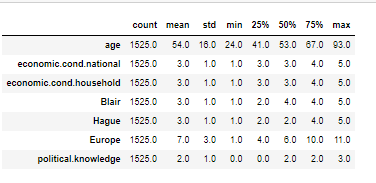
Most of the variables are int64 data type except vote and gender. These are object data type.

**Checking null values:**

****

**There are no null values in the data set.**

**Summary statistics:**

****

**Based on above output:**

**Insights:**

* **The minimum age of voters is 24 and max age is 93 years.**
* **50 % of the voters are below 53 years.**
* **75% o the voters have a positive assessment of economic condition.national rated 3.**
* **25% of voters only rated less than 3.**
* **The mean household economic condition assessed by the voters is rated to be 3 out of 5.**
* **75 % of voters have rated the household economic condition 3 out of 5.**
* **The mean rating given to the labour leader is 3 out of 5.**
* **The mean rating given to the conservative leader is almost 3 out of 5.**
* **50% voters have rated the leader less than 2.**
* **The mean score towards European integration is 7.**
* **The mean and median rating are 1.54 and 2 out of 3 respectively.**
  1. **Perform EDA (Check the null values, Data types, shape, Univariate, bivariate analysis). Also check for outliers (4 pts). Interpret the inferences for each (3 pts) Distribution plots(histogram) or similar plots for the continuous columns. Box plots, Correlation plots. Appropriate plots for categorical variables. Inferences on each plot. Outliers proportion should be discussed, and inferences from above used plots should be there. There is no restriction on how the learner wishes to implement this but the code should be able to represent the correct output and inferences should be logical and correct.**

**Categorical variables:**

['vote', 'gender', 'economic.cond.national', 'economic.cond.household', 'Blair', 'Hague', 'Europe', 'political.knowledge']

Numeric variable:

['age']

## Univariate Analysis for categorical variable

vote No of Levels: 2

Labour 1063

Conservative 462

Name: vote, dtype: int64

gender No of Levels: 2

female 812

male 713

Name: gender, dtype: int64

economic.cond.national No of Levels: 5

3 607

4 542

2 257

5 82

1 37

Name: economic.cond.national, dtype: int64

economic.cond.household No of Levels: 5

3 648

4 440

2 280

5 92

1 65

Name: economic.cond.household, dtype: int64

Blair No of Levels: 5

4 836

2 438

5 153

1 97

3 1

Name: Blair, dtype: int64

Hague No of Levels: 5

2 624

4 558

1 233

5 73

3 37

Name: Hague, dtype: int64

Europe No of Levels: 11

11 338

6 209

3 129

4 127

5 124

8 112

9 111

1 109

10 101

7 86

2 79

Name: Europe, dtype: int64

political.knowledge No of Levels: 4

2 782

0 455

3 250

1 38

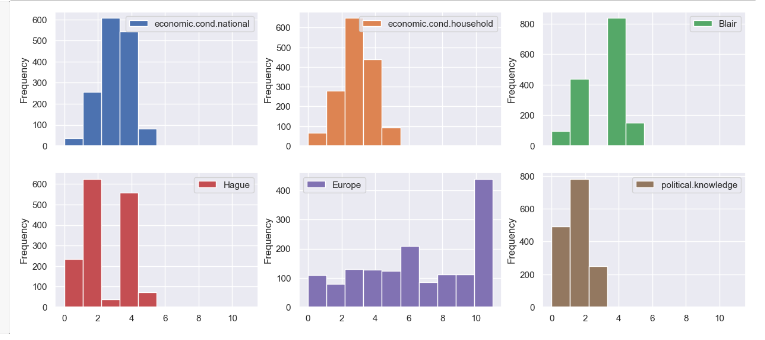
Name: political.knowledge, dtype: int64

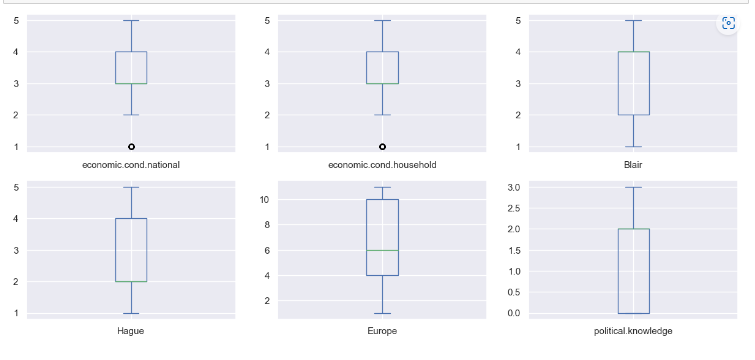
## checking target variable proportion

Labour 69.704918

Conservative 30.295082

Name: vote, dtype: float64

****

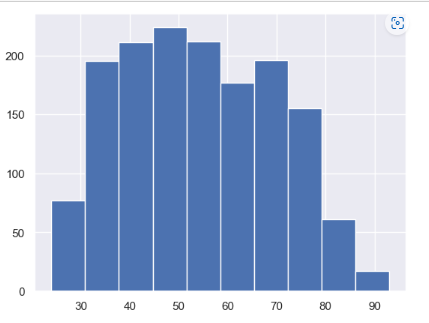
****

**Based on above output:**

**Insights:**

* National economic condition: outliers are present in this variable. The data distribution have cannot be fully normal due to ordinal in nature.
* National household condition: outliers are present in this variable. The data distribution is not normal due to ordinal data nature.
* Blair: There are no outliers in this variable and data distribution also not in normal.
* Hauge: There are no outliers in this variable and data distribution also not in normal due ordinal data.
* Europe: There are no outliers in this variable and data distribution also not in normal due ordinal data.
* Political knowledge: There are no outliers in this variable and data distribution also not in normal due ordinal data.

**Univariate Analysis for numeric variable:**

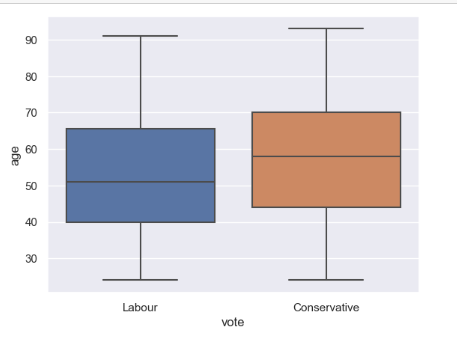
****

****

**Age:** The data distribution is nearly normally distributed. The distribution has negligible skewness and no outliers found.

## Bivariate and Multivariate Analysis:

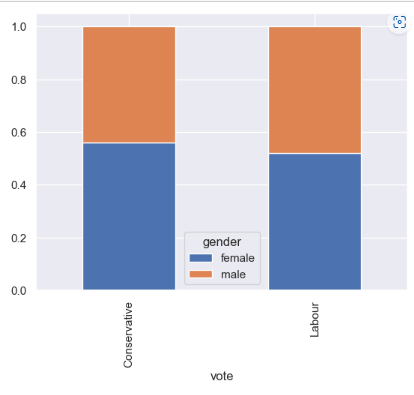
## Target Variable vs Numeric Independent Variables:



**Insights:**

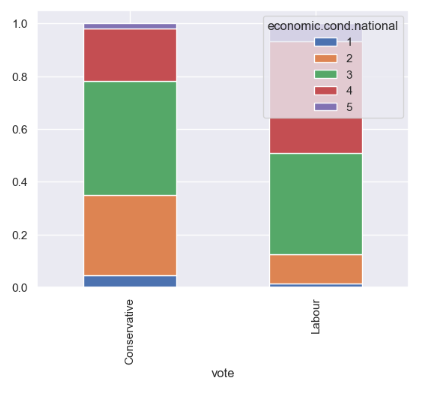
* The voter’s age group between 40 and 65 voted to labour party.
* The voter’s age group between 45 and 70 voted to conservative party.

## Target Variable vs Categorical Independent Variables



**Insights**:

* Almost 60% of females and 40 % males voted for conservative party.
* Almost 50% of females and 50 % males voted for conservative party.

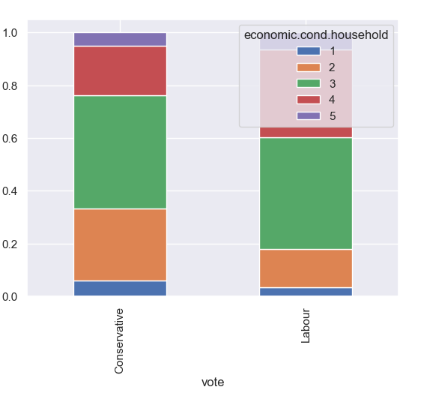


**Insights:**

**Almost 42% of labour voters have rated a high score of 4 out of 5, whereas only 20% of conservative voters have rated the high score 0f 4 out of 5.**

**Only 2% of conservative voters have rated perfect 5 and 7% of labour voters have rated perfect 5.**

**30% of conservative voters have rated an average score of 3.**

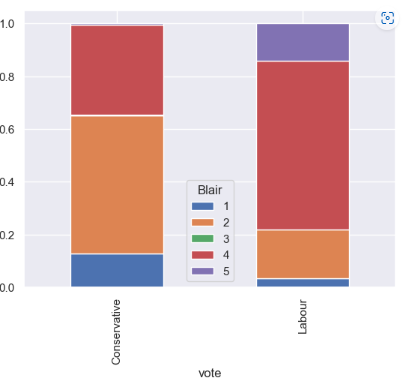
:

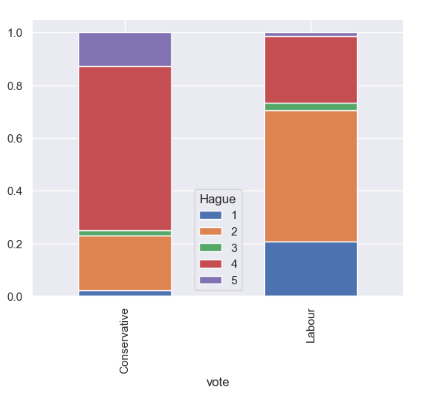
**Insights**:

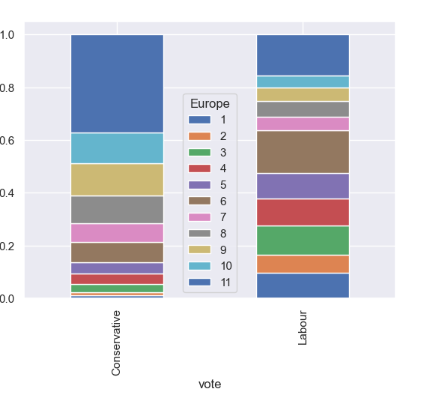
33% of labour voters have rated a high score of 4 out of 5

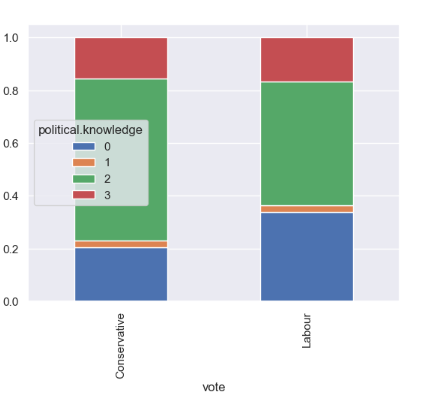
19% of conservative voters have rated the economic household condition as 4.

The majority of conservative voters and labour voters indicated that the household economic condition is average, rated at 3 out of 5.

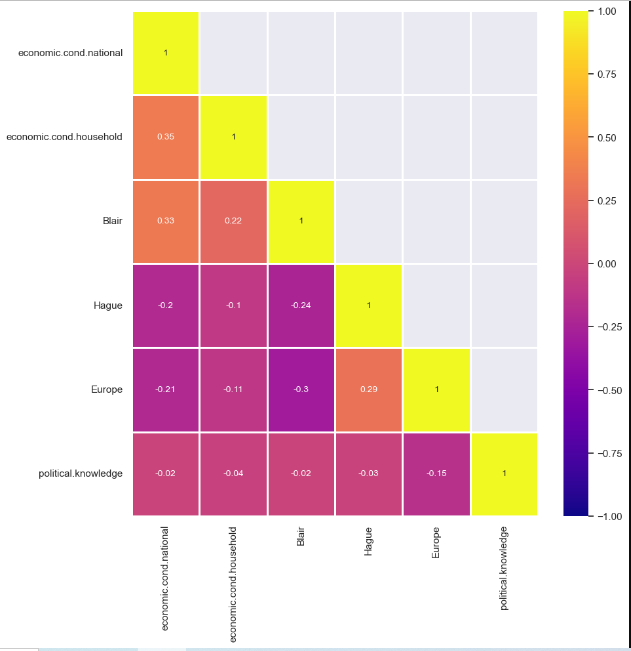


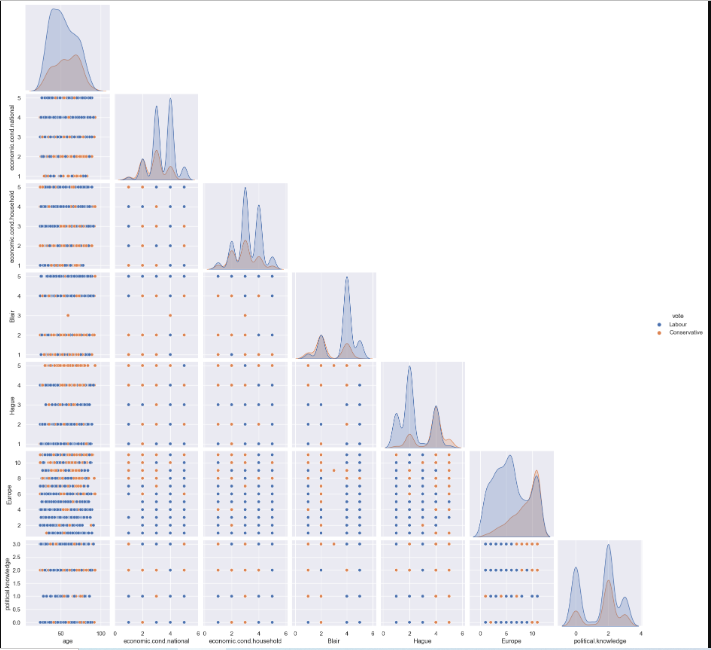
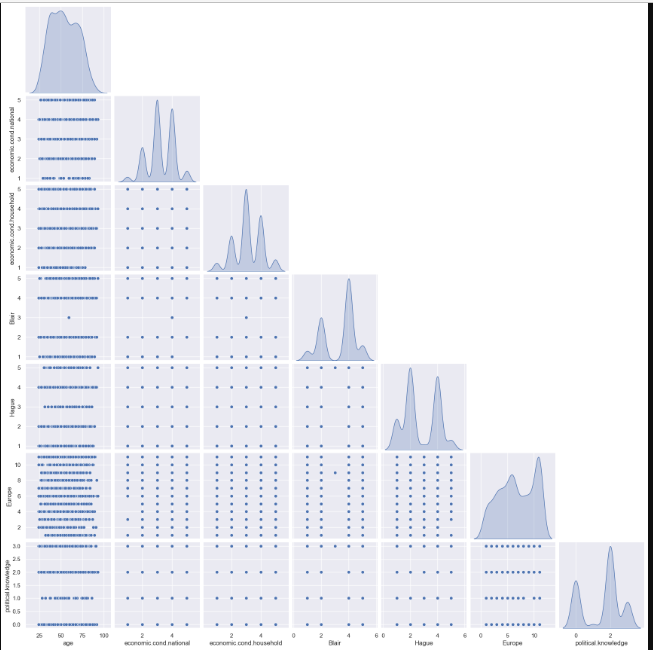






**Correlation Plot:**



**Pair plot**

* 1. **Encode the data (having string values) for Modelling. Is Scaling necessary here or not?( 2 pts), Data Split: Split the data into train and test (70:30) (2 pts). The learner is expected to check and comment about the difference in scale of different features on the bases of appropriate measure for example std dev, variance, etc. Should justify whether there is a necessity for scaling. Object data should be converted into categorical/numerical data to fit in the models. (pd.categorical().codes(), pd.get\_dummies(drop\_first=True)) Data split, ratio defined for the split, train-test split should be discussed.**
* Vote column consists two characters, Labour & Conservative.
* Gender column consists of object type data so we encode the "gender" variable by using pd.get\_dummies() and dropping the first column to make sure we are not affected by dummies. Also we will replace vote columns characters by 0 & 1 for labour and conservative respectively. Below fig shows after applying the dummies and replacement method.

**Before encoing:**

# 

# After encoding:

# 

**Scaling**

We are not going to scale the data for Logistic regression, LDA and Naive Bayer’s models as it is not necessary. But in case of KNN, it is necessary to scale

The data, it is a distance-based algorithm (typically based on Euclidean distance). Scaling the data gives similar weightage to all the variables. We use Min max scaler().

## Checking Target Variable class proportion

**Y\_train.value counts**

1 68.884724

0 31.115276

Name: vote\_Labour, dtype: float64

**y\_test.value counts**

1 0.716157

0 0.283843

Name: vote\_Labour, dtype: float64

**Train-Test Split**

We split the scaled data set into Train and Test 70% & 30% respectively and stored in separate variable which we used for only KNN and SVM. We considered random\_state=1 for reproducible results.

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

**Logistic Regression Model:**

**Prediction on Train & Test Dataset**

**Class Label Prediction**

**Ytrain\_predict**

array([0, 1, 1, 1, 1, 1, 0, 1, 1, 1], dtype=uint8)

**ytest\_predict**

array([0, 0, 1, 1, 1, 0, 1, 1, 1, 1], dtype=uint8)

**Class Probability Prediction**

# 

# Model Evaluation

**Training Data**

**Accuracy score:** 0.8406747891283973

**Recall score:** 0.9074829931972789

**Precision score:** 0.8673602080624188

**F1 score:** 0.886968085106383

**AUC and ROC for the training data**

# 

# Confusion Matrix & Classification Report Metrics

# 

# 

**Test Data**

**Accuracy score:** 0.8231441048034934

**Recall score:** 0.8902439024390244

**Precision score:** 0.8664688427299704

**F1 score:** 0.8781954887218046

**AUC and ROC**

# 

# Confusion Matrix & Classification Report Metrics

# 

# Linear Discriminant Analysis

**Prediction**

**class labels prediction on train & test data and save it**

**pred\_train = lda.predict(X\_train)**

**pred\_test = lda.predict(X\_test)**

**Model Evaluation**

**Training Data**

**ytrain\_predict = lda.predict(X\_train)**

**ytest\_predict = lda.predict(X\_test)**

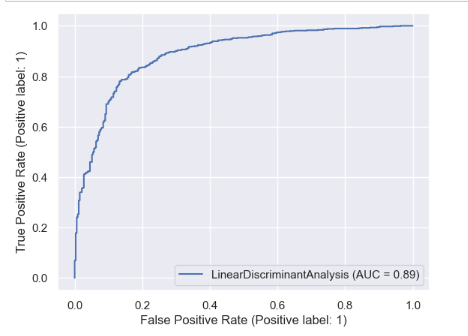
**Accuracy score:** 0.8369259606373008

**Recall score:** 0.8979591836734694

**Precision score:** 0.8695652173913043

**F1 score:** 0.8835341365461847

**AUC and ROC**

****

# 

# Test Data

**Accuracy score:** 0.8187772925764192

**Recall score:** 0.8810975609756098

**Precision score:** 0.8678678678678678

**F1 score:** 0.8744326777609682

# 

**Naive Bayes Model**

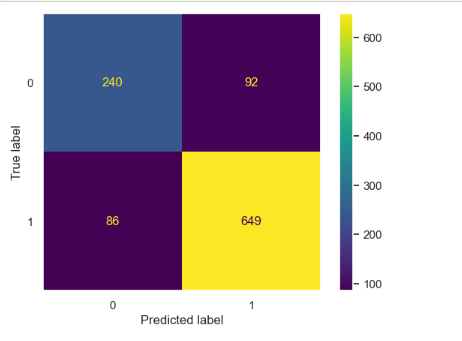
**Training Performance:**

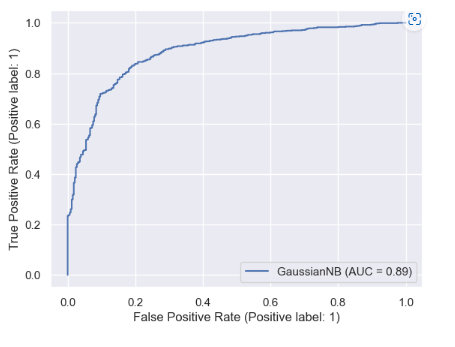
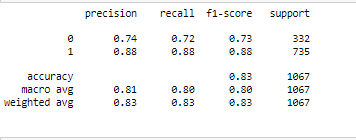
**Accuracy score:** 0.8331771321462043

**Recall score:** 0.8829931972789116

**Precision score:** 0.8758434547908233

**F1 score:** 0.8794037940379404

****

****

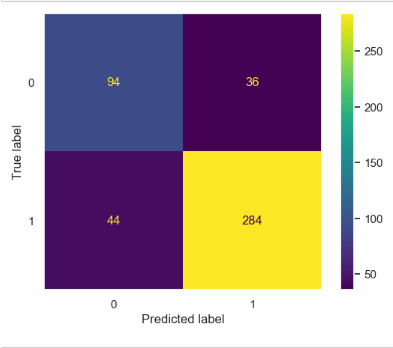
**Test Performance**

**Accuracy score:** 0.8253275109170306

**Recall score:** 0.8658536585365854

**Precision score:** 0.8875

**F1 score:** 0.8765432098765432

****

# 

**KNN Model**

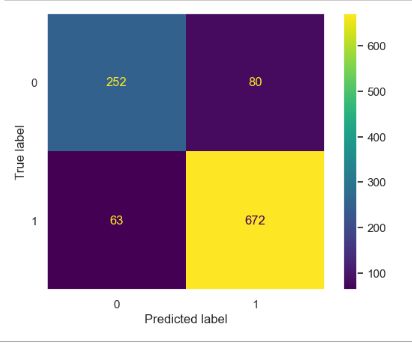
**Training Performance**

**Accuracy score:** 0.865979381443299

**Recall score:** 0.9142857142857143

**Precision score:** 0.8936170212765957

**F1 score:** 0.9038332212508405



# 

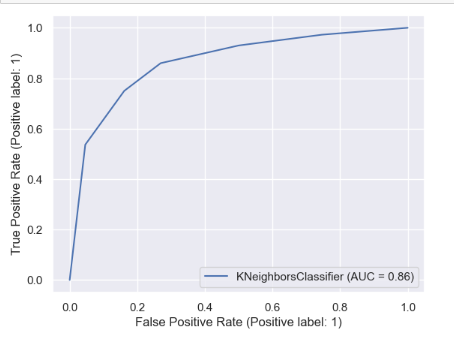
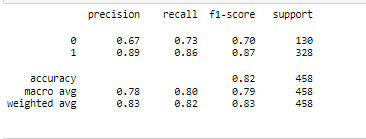
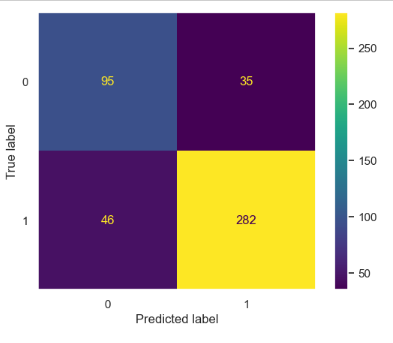
# Test Performance

**Accuracy score:** 0.8231441048034934

**Recall score:** 0.8597560975609756

**Precision score:** 0.889589905362776

**F1 score:** 0.8744186046511628

****

**KNN model 2**

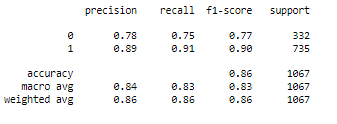
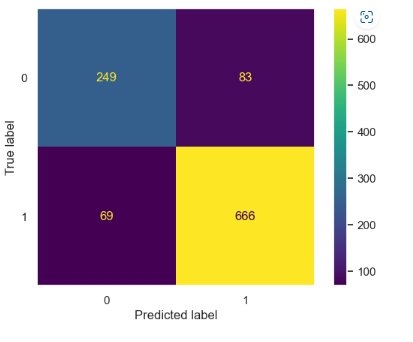
**Training Performance**

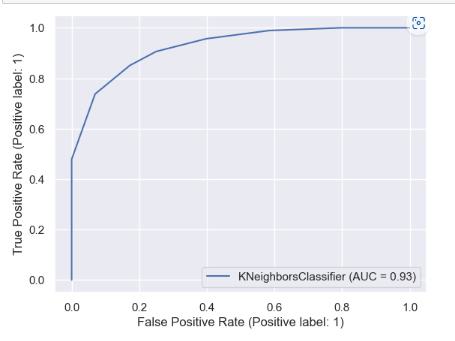
**Accuracy score:** 0.8575445173383318

**Recall score:** 0.9061224489795918

**Precision score:** 0.8891855807743658

**F1 score:** 0.8975741239892183

****

****

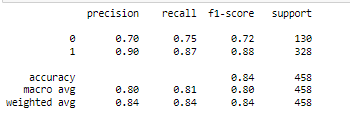
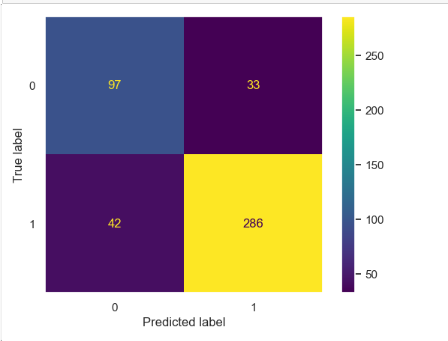
**Test Performance**

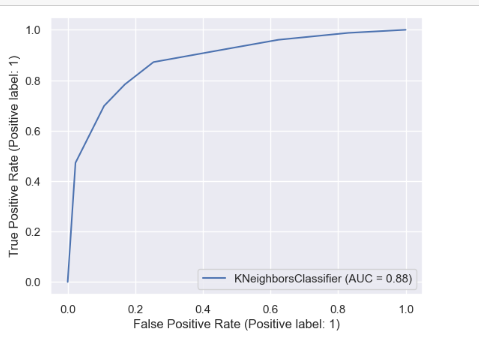
**Accuracy score:** 0.8362445414847162

**Recall score:** 0.8719512195121951

**Precision score:** 0.896551724137931

**F1 score:** 0.8840803709428129

****

****

**KNN model 3**

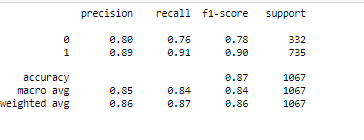
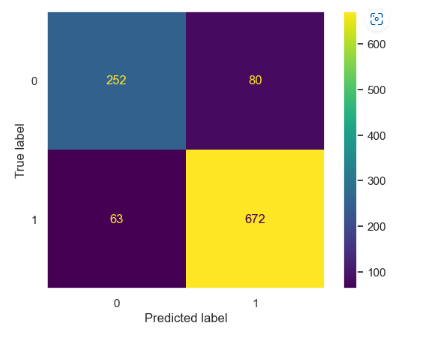
**Training Performance**

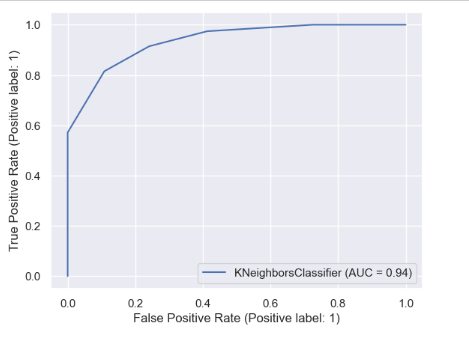
**Accuracy score:** 0.865979381443299

**Recall score:** 0.9142857142857143

**Precision score:** 0.8936170212765957

**F1 score:** 0.9038332212508405

****

****

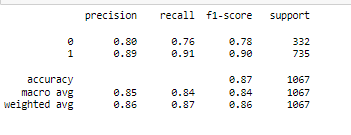
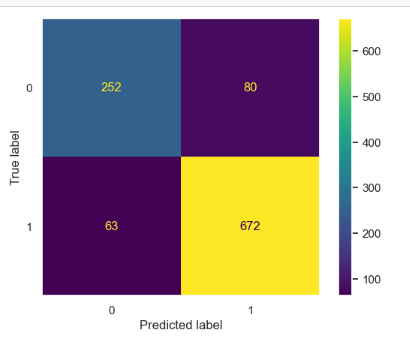
**Test Performance**

**Accuracy score:** 0.8231441048034934

**Recall score:** 0.8597560975609756

**Precision score:** 0.889589905362776

**F1 score:** 0.8744186046511628

****

# 

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

**Bagging**

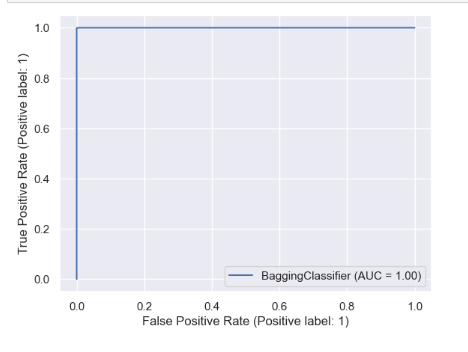
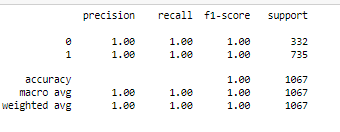
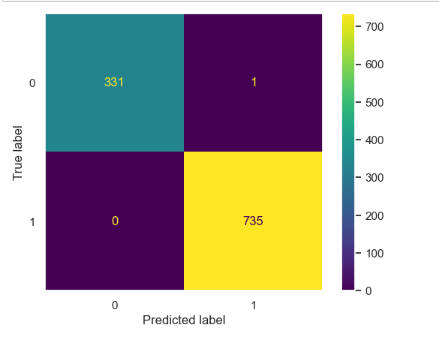
**Training Performance**

**Accuracy score:** 0.9990627928772259

**Recall score:** 1.0

**Precision score:** 0.998641304347826

**F1 score:** 0.9993201903467028

****

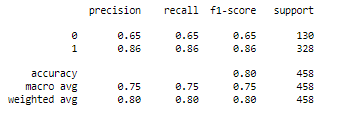
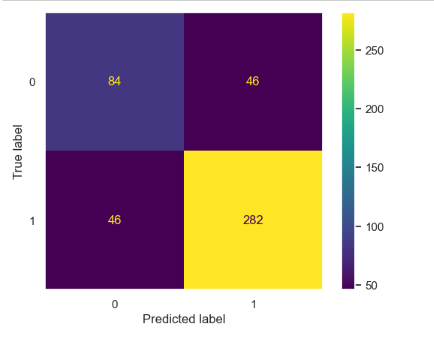
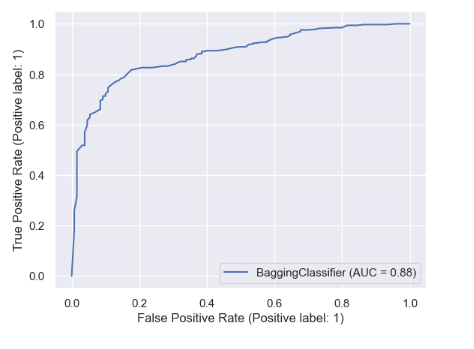
**Test Performance**

**Accuracy score:** 0.7991266375545851

**Recall score:** 0.8597560975609756

**Precision score:** 0.8597560975609756

**F1 score:** 0.8597560975609756

**  **

**Ada Boost**

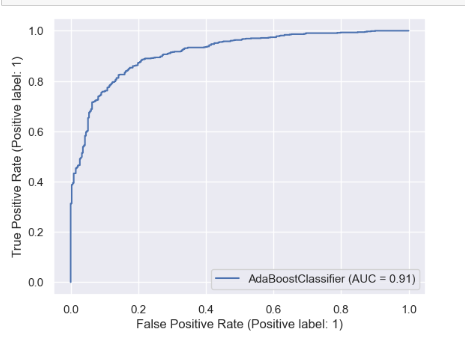
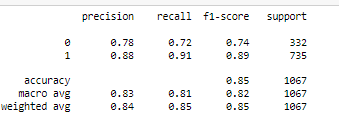
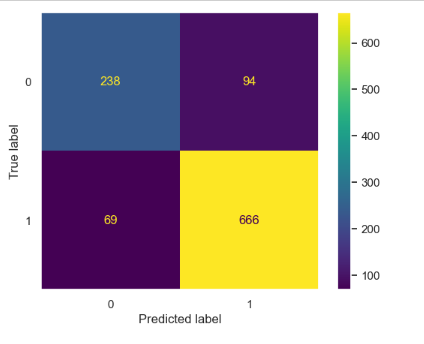
**Training Performance**

**Accuracy score:** 0.8472352389878163

**Recall score:** 0.9061224489795918

**Precision score:** 0.8763157894736842

**F1 score:** 0.8909698996655518

****

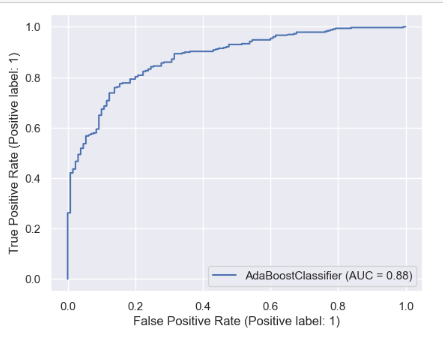
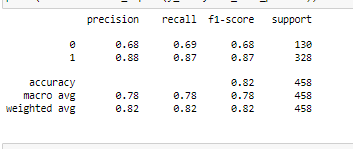
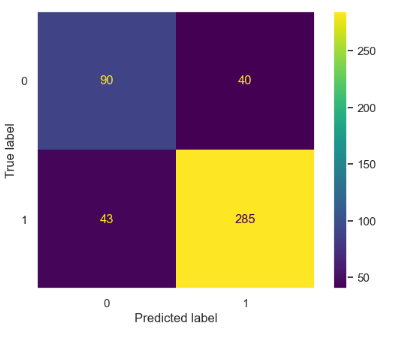
**Test Performance**

**Accuracy score:** 0.8187772925764192

**Recall score:** 0.8689024390243902

**Precision score:** 0.8769230769230769

**F1 score:** 0.8728943338437979

****

**Gradient Boosting**

**Training Performance**

**Accuracy score:** 0.8865979381443299

**Recall score:** 0.9306122448979591

**Precision score:** 0.9071618037135278

**F1 score:** 0.918737407656145

# 

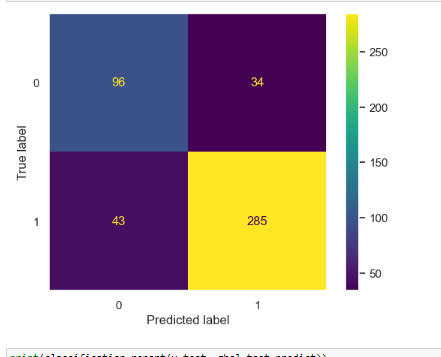
# Test Performance

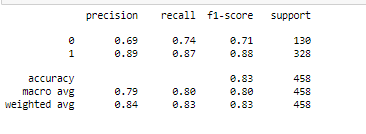
**Accuracy score:** 0.8318777292576419

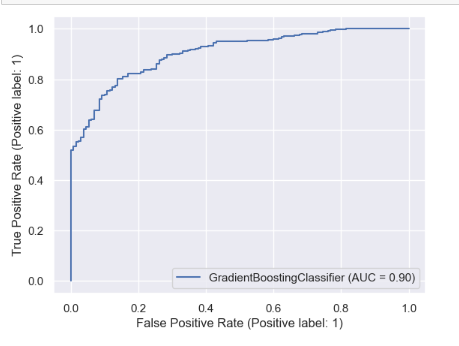
**Recall score:** 0.8689024390243902

**Precision score:** 0.8934169278996865

**F1 score:** 0.8809891808346213

****

****

****

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

**Model tunning:**

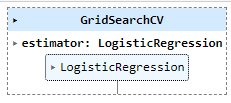
**GridSearchCV for logistic regression:**

**Parameters:**

Grid= {'penalty':['l2','none','l1','elasticnet'],

'solver':['sag','lbfgs','saga','newton-cg','liblinear'],

'tol':[0.0001,0.00001],'l1\_ratio':[0.25,0.5,0.75]}



**Grid search best parameters:**

{'l1\_ratio': 0.25, 'penalty': 'l1', 'solver': 'liblinear', 'tol': 0.0001}

**Grid search best parameters:**

LogisticRegression(l1\_ratio=0.25, max\_iter=10000, n\_jobs=2, penalty='l1',

random\_state=1, solver='liblinear')

## Train dataset:

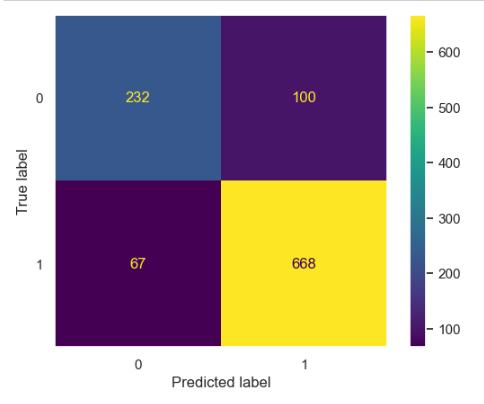
**Accuracy score:** 0.8406747891283973

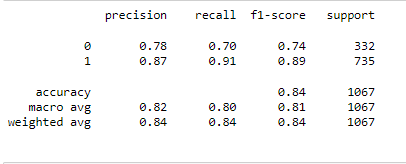
**Recall score:** 0.8965986394557823

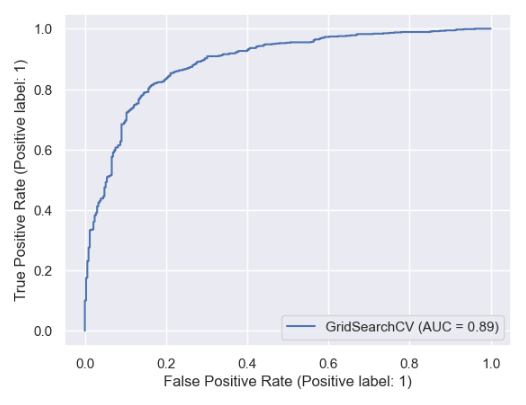
**Precision score:** 0.8751660026560425

**F1 score:** 0.885752688172043

**Confusion matrix:**







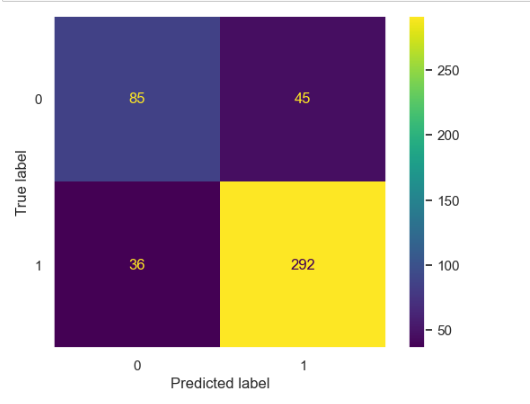
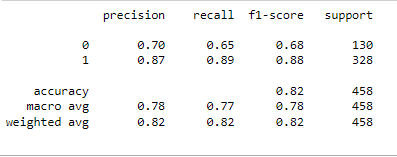
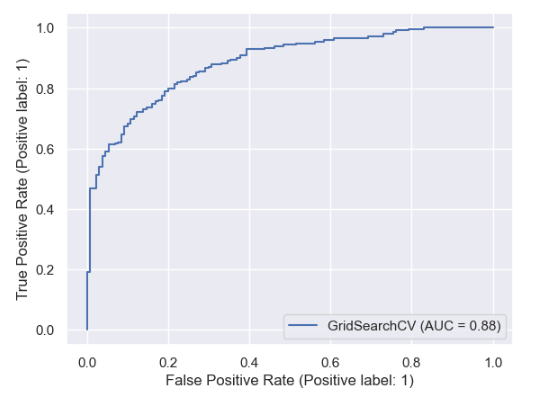
**Test dataset:**

**Accuracy score:** 0.8231441048034934

**Recall score:** 0.8780487804878049

**Precision score:** 0.8753799392097265

**F1 score:** 0.8767123287671232

**GridSearchCV on Linear discriminant analysis:**

**Parameters:**

paramgrid = {'n\_components':['int',None],

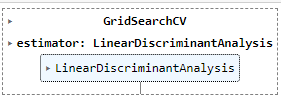
'priors':['array-like of shape',None],

'shrinkage':[None,'auto' or 'float'],

'solver':['svd','lsqr','eigen'],

'store\_covariance':['bool',False],

'tol':['float',0.0001]}



**Best parameters:**

{'n\_components': None, 'priors': None, 'shrinkage': 'auto', 'solver': 'lsqr', 'store\_covariance': 'bool', 'tol': 'float'}

**Best estimator:**

LinearDiscriminantAnalysis(shrinkage='auto', solver='lsqr',

store\_covariance='bool', tol='float')

**Train dataset:**

**Accuracy score:** 0.8406747891283973

**Recall score:** 0.8965986394557823

**Precision score:** 0.8751660026560425

**F1 score:** 0.885752688172043

**Test dataset:**

**Accuracy score:** 0.8231441048034934

**Recall score:** 0.8780487804878049

**Precision score:** 0.8753799392097265

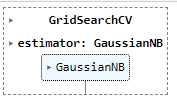
**F1 score:** 0.8767123287671232

**GridSearchCV on Navie bayes:**

**Parameters:**

paramgrid = {'priors':[None,'array-like of shape'],

'var\_smoothing':[1e-09, float]}

****

**Best parameters:**

{'priors': None, 'var\_smoothing': 1e-09}

**Best estimator:**

GaussianNB()

**Train dataset:**

**Accuracy score:** 0.8331771321462043

**Recall score:** 0.8829931972789116

**Precision score:** 0.8758434547908233

**F1 score:** 0.8794037940379404

**Test dataset:**

**Accuracy score:** 0.8231441048034934

**Recall score:** 0.8780487804878049

**Precision score:** 0.8753799392097265

**F1 score:** 0.8767123287671232

**GridSearchCV on KNN:**

**Parameters:**

paramgrid = { 'n\_neighbors':[5,'int'],

'weights':['uniform','distance'],

'algorithm':['auto','ball\_tree', 'kd\_tree', 'brute'],

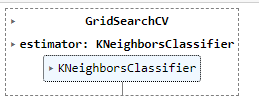
'leaf\_size':[30,'int'],

'p':[2,'int'],

'metric':['minkowski','str or callable'],

'metric\_params':[None,'dict'],

'n\_jobs':[None,'int']}



**Best parameters:**

{'algorithm': 'ball\_tree', 'leaf\_size': 30, 'metric': 'minkowski', 'metric\_params': None, 'n\_jobs': None, 'n\_neighbors': 5, 'p': 2, 'weights': 'uniform'}

**Best estimator:**

KNeighborsClassifier(algorithm='ball\_tree')

**Train dataset:**

**Accuracy score:** 0.8650421743205249

**Recall score:** 0.9129251700680272

**Precision score:** 0.8934753661784287

**F1 score:** 0.9030955585464334

**Test dataset:**

**Accuracy score:** 0.8231441048034934

**Recall score:** 0.8780487804878049

**Precision score:** 0.8753799392097265

**F1 score:** 0.8767123287671232

**GridSearchCV on bagging:**

**Parameters:**

paramgrid = {'base\_estimator':[None,'object','cart'],

'n\_estimators':[10,100,1000],

'max\_samples':[1.0,'int'or'float'],

'max\_features':[1.0,'int','float'],

'bootstrap':[True,'bool'],

'bootstrap\_features':[False,'bool'],

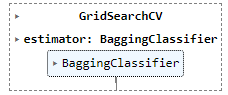
'oob\_score':[False,'bool'],

'warm\_start':[False,'bool'],

'n\_jobs':[None,'int'],

'random\_state':[None],

'verbose':[0,'int']}



**Best parameters**:

{'base\_estimator': None, 'bootstrap': True, 'bootstrap\_features': 'bool', 'max\_features': 1.0, 'max\_samples': 1.0, 'n\_estimators': 1000, 'n\_jobs': None, 'oob\_score': 'bool', 'random\_state': None, 'verbose': 0, 'warm\_start': False}

**Best estimator:**

BaggingClassifier(bootstrap\_features='bool', n\_estimators=1000,

oob\_score='bool')

**Train dataset:**

**Accuracy score:** 0.993439550140581

**Recall score:** 0.998639455782313

**Precision score:** 0.9918918918918919

**F1 score:** 0.9952542372881357

**Test dataset:**

**Accuracy score:** 0.8231441048034934

**Recall score:** 0.8780487804878049

**Precision score:** 0.8753799392097265

**F1 score:** 0.8767123287671232

**GridSearchCV on Adaboosting:**

**Parameters:**

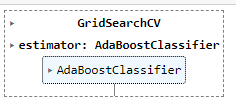
paramgrid = { 'base\_estimator':[None,'object'],

'n\_estimators':[50,100,1000],

'learning\_rate':[1.0,'float'],

'algorithm':['SAMME.R','SAMME'],

'random\_state':[None,int]}



**Best parameters:**

{'algorithm': 'SAMME', 'base\_estimator': None, 'learning\_rate': 1.0, 'n\_estimators': 100, 'random\_state': None}

**Best estimators:**

AdaBoostClassifier(algorithm='SAMME', n\_estimators=100)

**Train dataset:**

**Accuracy score:** 0.8406747891283973

**Recall score:** 0.8965986394557823

**Precision score:** 0.8751660026560425

**F1 score:** 0.885752688172043

**Test dataset:**

**Accuracy score:** 0.8231441048034934

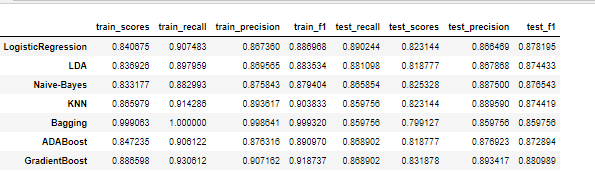
**Recall score:** 0.8780487804878049

**Precision score:** 0.8753799392097265

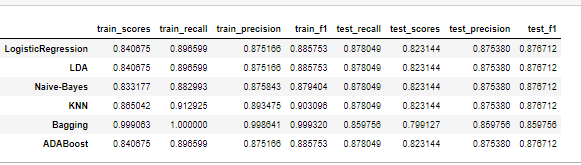
**F1 score:** 0.8767123287671232

* 1. **Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC\_AUC score for each model, classification report (4 pts). Final Model - Compare and comment on all models on the basis of the performance metrics in a structured tabular manner. Describe on which model is best/optimized, After comparison which model suits the best for the problem in hand on the basis of different measures. Comment on the final model.(3 pts)**

**Comparison of Different Models:**

****

**After model tunning comparision of models:**

****

On the in depth observation from the different models used in this case, the data has been inferred that in this case KNN model with n=7 is highly optimized as compared to the other models, after making in depth comparisons of accuracy, recall, model score, and AUC score of training and test data of different models.

* 1. **Based on your analysis and working on the business problem, detail out appropriate insights and recommendations to help the management solve the business objective. There should be at least 3-4 Recommendations and insights in total. Recommendations should be easily understandable and business specific, students should not give any technical suggestions. Full marks should only be allotted if the recommendations are correct and business specific.**

**Inferences:**

* The overall data has needed scaling in order to make uniform for the data analysis.
* There are outliers being present in some variable.
* The overall training and testing of this dataset using different methods has given similar results which is clearly showing that the overall data modeling, model tuning and scaling has been done properly.
* Bagging has exhibited big difference in the training and testing data, rest others have exhibited almost similar or very small gap between testing and training.
* As per the vote count of the survey data, labour party has achieved 1063 votes and conservative party has achieved 462 votes which is even less than half of the votes achieved by labour party.

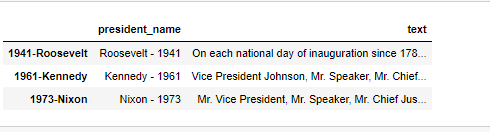
**Problem 2:**

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

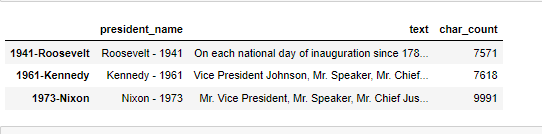
1. President Franklin D. Roosevelt in 1941
2. President John F. Kennedy in 1961
3. President Richard Nixon in 1973

**2.1) Find the number of characters, words and sentences for the mentioned documents. (Hint: use .words (), .raw(), .sent() for extracting counts)**

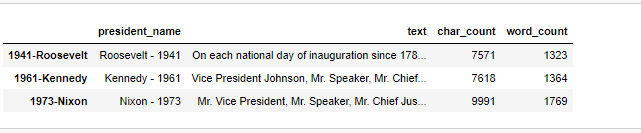
**Sample of data:**

****

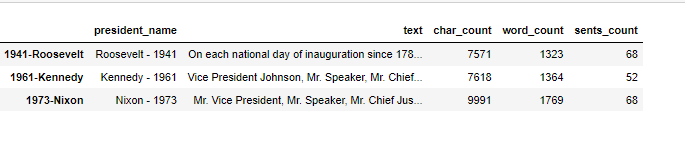
**Number of characters:**

****

**Number of words:**

****

**Number of sentences:**

****

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

**Before removing stopwords:**

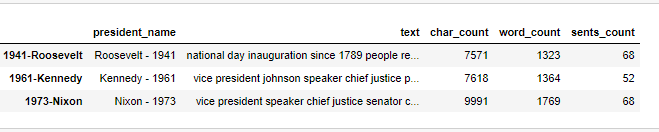
1941-Roosevelt on each national day of inauguration since 178...

1961-Kennedy vice president johnson mr speaker mr chief jus...

1973-Nixon mr vice president mr speaker mr chief justice ...

Name: text, dtype: object

**After removing stopwords:**

****

**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)**

**Most repeated word in speeches:**

nation 11

know 10

spirit 9

dtype: int64

**top three words:**

world 8

sides 8

new 7

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

