Machine Learning Project

Business Report – Predictive Modeling Project

KARUNA_ML_CODED_PROJECT_20-1-2024

KARUNA PASHTE

INDEX

Problem No.1

Context

CNBE, a prominent news channel, is gearing up to provide insightful coverage of recent elections, recognizing the importance of data-driven analysis. A comprehensive survey has been conducted, capturing the perspectives of 1525 voters across various demographic and socio-economic factors. This dataset encompasses 9 variables, offering a rich source of information regarding voters' characteristics and preferences.

Objective

The primary objective is to leverage machine learning to build a predictive model capable of forecasting which political party a voter is likely to support. This predictive model, developed based on the provided information, will serve as the foundation for creating an exit poll. The exit poll aims to contribute to the accurate prediction of the overall election outcomes, including determining which party is likely to secure the majority of seats.

Data Description

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.

QUESTION 1. Define the problem and perform Exploratory Data Analysis.

Answer:

1.1 Import the dataset using head function and understand the problem.

	Unnamed: 0	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	1	Labour	43	3	3	4	1	2	2	female
1	2	Labour	36	4	4	4	4	5	2	male
2	3	Labour	35	4	4	5	2	3	2	male
3	4	Labour	24	4	2	2	1	4	0	female
4	5	Labour	41	2	2	1	1	6	2	male
5	6	Labour	47	3	4	4	4	4	2	male
6	7	Labour	57	2	2	4	4	11	2	male
7	8	Labour	77	3	4	4	1	1	0	male
8	9	Labour	39	3	3	4	4	11	0	female
9	10	Labour	70	3	2	5	1	11	2	male

1.2 Dropping the unnamed column as it is not that important.

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

1.3.How many rows and columns present in the dataset:

There are 1525 rows and 9 columns in the dataset are available.

1.4.Checking null values and duplicate values in dataset:

```
vote                       0
age                        0
economic.cond.national     0
economic.cond.household    0
Blair                      0
Hague                      0
Europe                     0
political.knowledge        0
gender                     0
dtype: int64
```

There are no null values present in dataset.
There are 8 duplicate values present in the dataset.

1.5 Check the basic info about the dataset :

```
<class 'pandas.core.frame.DataFrame'>
Index: 1517 entries, 0 to 1524
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype 
---  ------                   --------------  ----- 
 0   vote                     1517 non-null   object
 1   age                      1517 non-null   int64 
 2   economic.cond.national   1517 non-null   int64 
 3   economic.cond.household  1517 non-null   int64 
 4   Blair                    1517 non-null   int64 
 5   Hague                    1517 non-null   int64 
 6   Europe                   1517 non-null   int64 
 7   political.knowledge      1517 non-null   int64 
 8   gender                   1517 non-null   object
dtypes: int64(7), object(2)
memory usage: 118.5+ KB
```

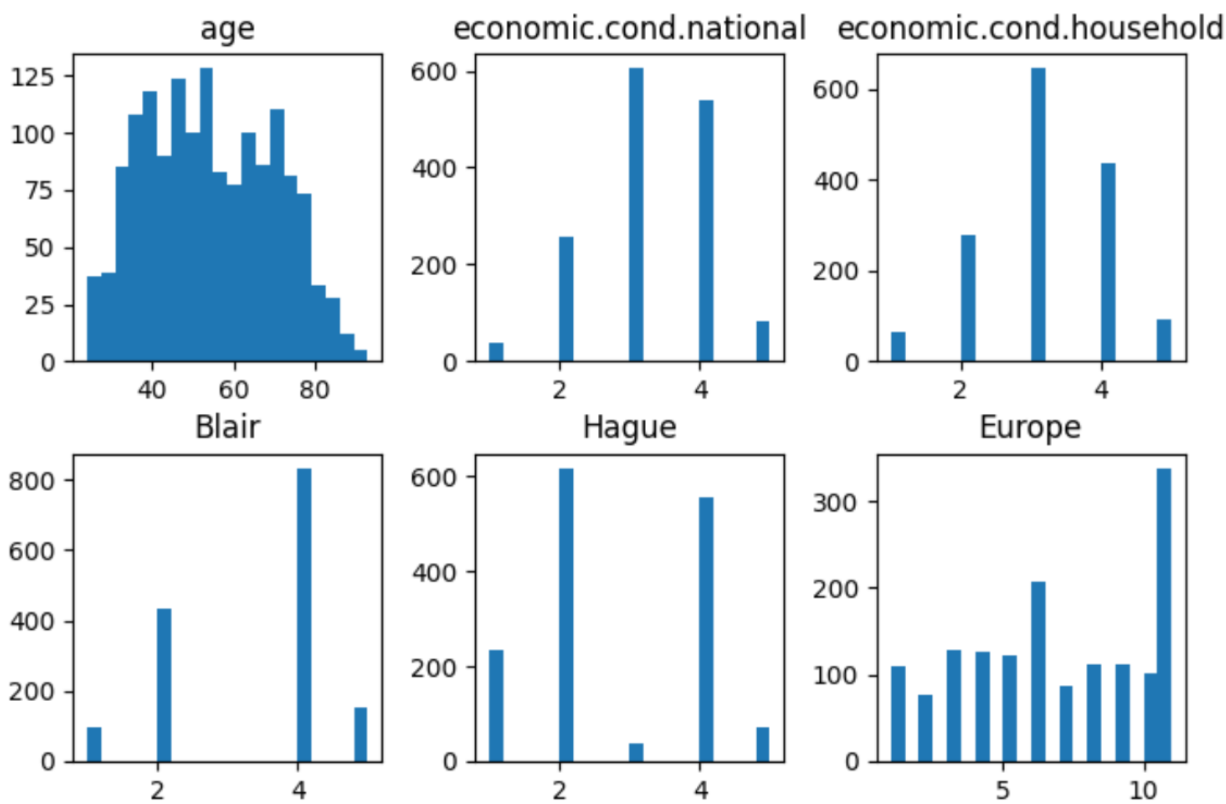
There are two variables with string values and remaining are in numerical.

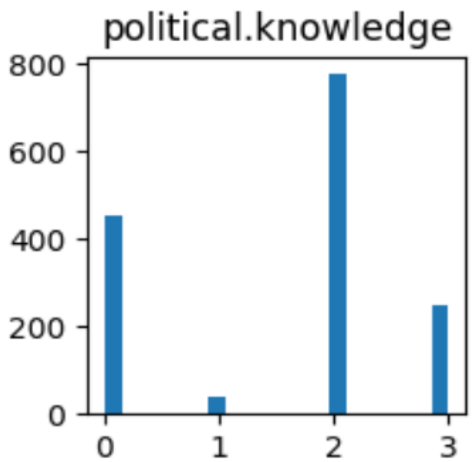
1.6 Check the statistical summary of dataset:

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
vote	1517	2	Labour	1057	NaN	NaN	NaN	NaN	NaN	NaN	NaN
age	1517.0	NaN	NaN	NaN	54.241266	15.701741	24.0	41.0	53.0	67.0	93.0
economic.cond.national	1517.0	NaN	NaN	NaN	3.245221	0.881792	1.0	3.0	3.0	4.0	5.0
economic.cond.household	1517.0	NaN	NaN	NaN	3.137772	0.931069	1.0	3.0	3.0	4.0	5.0
Blair	1517.0	NaN	NaN	NaN	3.335531	1.174772	1.0	2.0	4.0	4.0	5.0
Hague	1517.0	NaN	NaN	NaN	2.749506	1.232479	1.0	2.0	2.0	4.0	5.0
Europe	1517.0	NaN	NaN	NaN	6.740277	3.299043	1.0	4.0	6.0	10.0	11.0
political.knowledge	1517.0	NaN	NaN	NaN	1.540541	1.084417	0.0	0.0	2.0	2.0	3.0
gender	1517	2	female	808	NaN	NaN	NaN	NaN	NaN	NaN	NaN

There are no missing values or non numerical values present.
Most of the variables have equal mean and median.

1.7.Perform Exploratory Data Analysis:
Check the normal distribution of observation using a histogram.

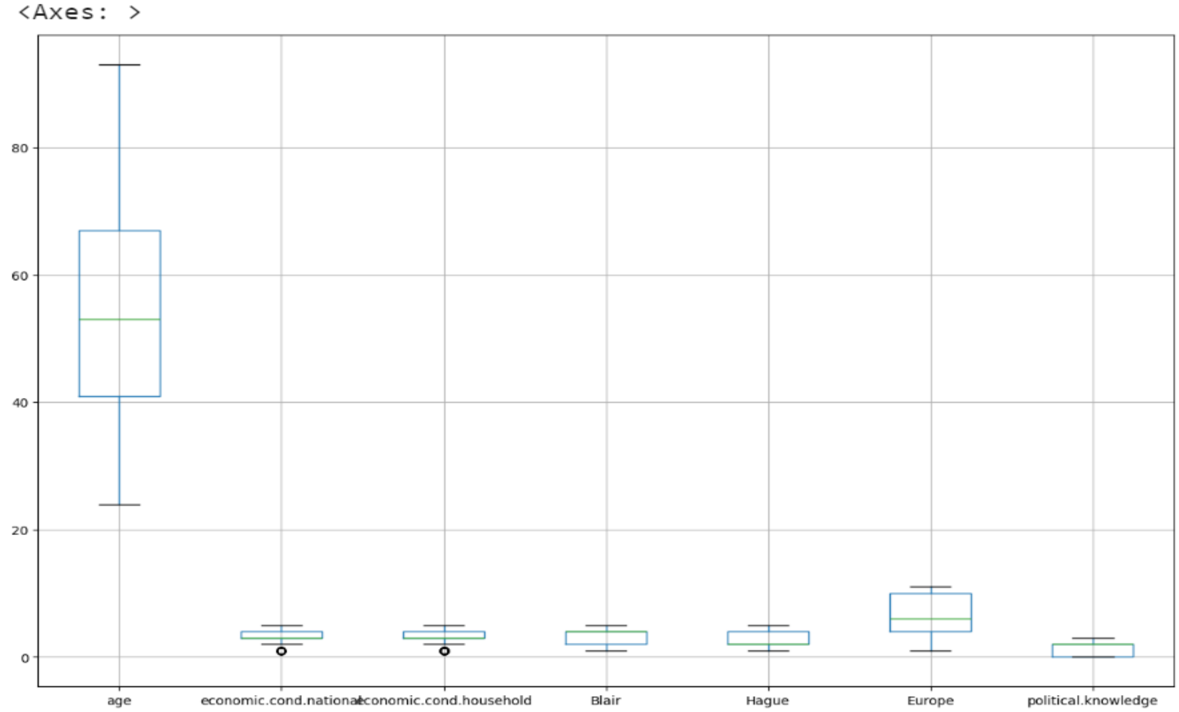




Age has high standard deviation from mean value.
Other than age most of the variables has less standard deviation from the mean value.

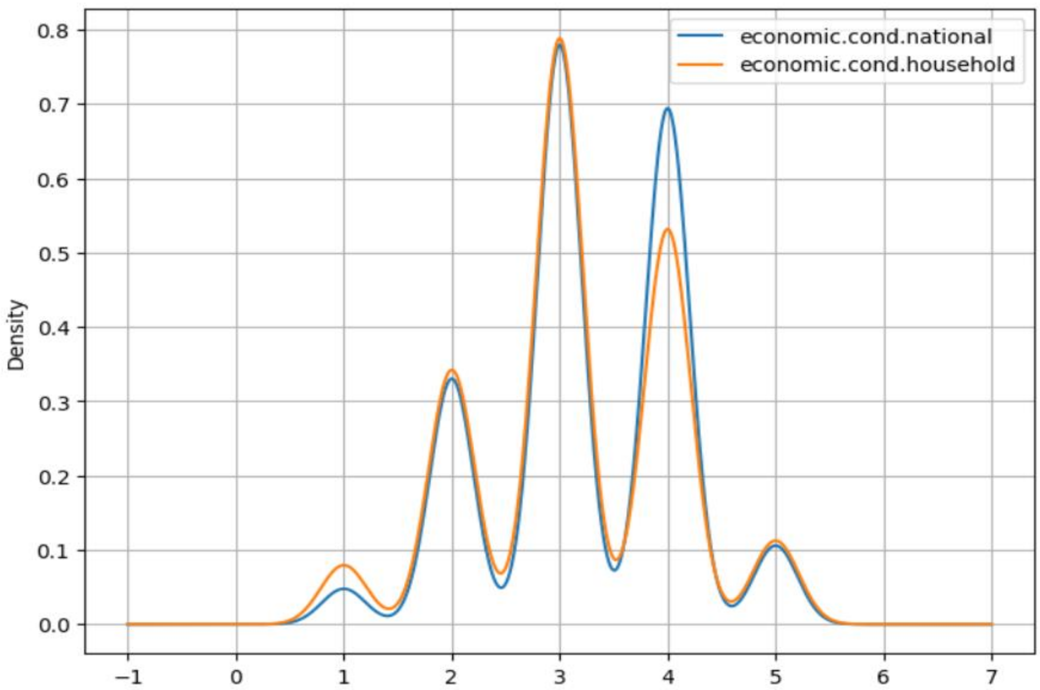
Question 2. Data Pre-processing:
2.1. Check outliers using boxplot:

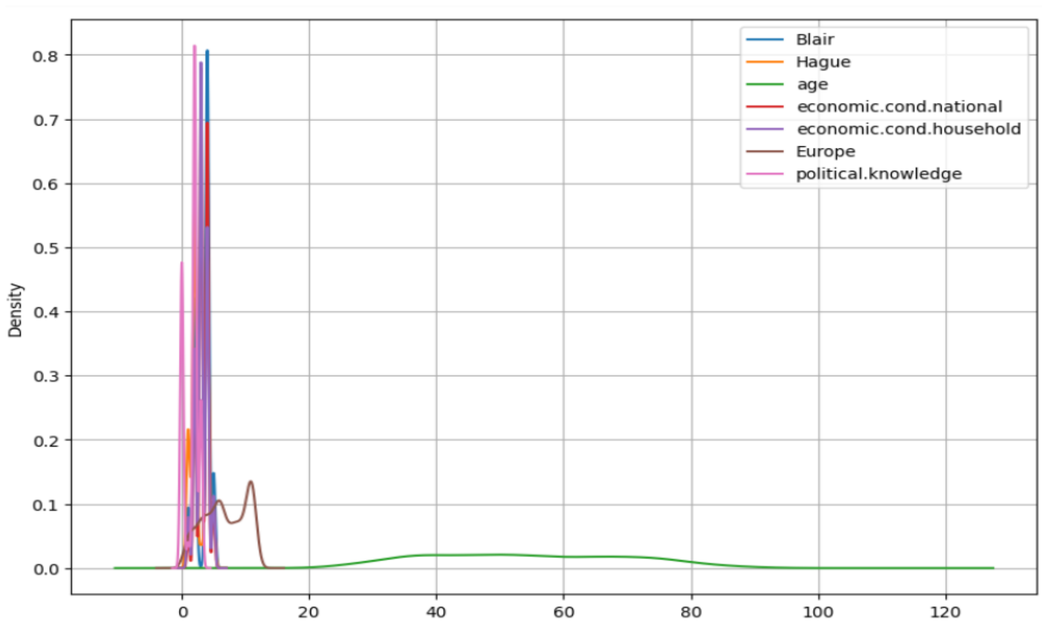
<Axes: >



Only two variables has outliers. Let us remove the outlier to treat the model.

2.2. Check the distribution of the variables:





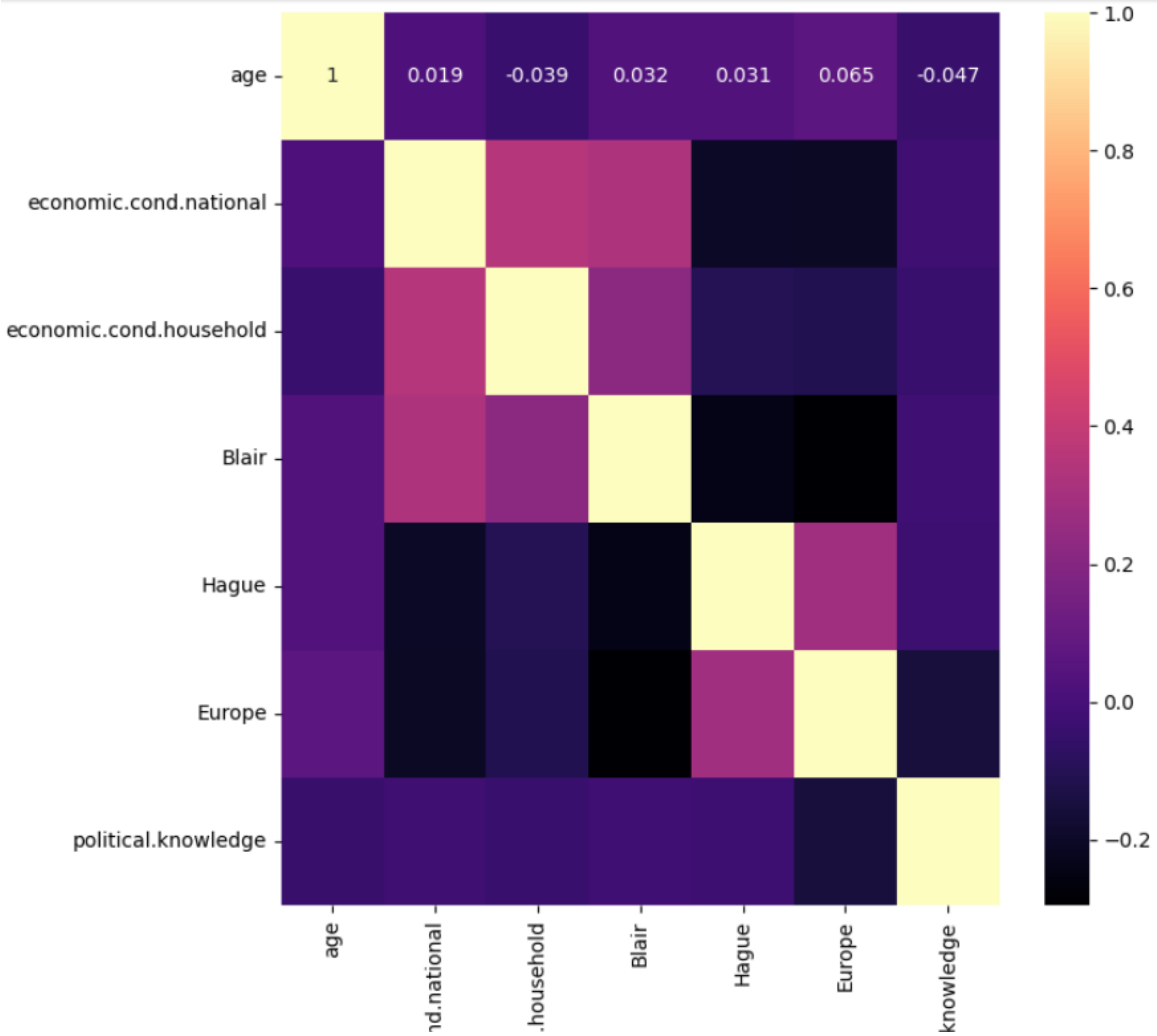
This are distribution of variables with outliers. Two variables are almost having the same distribution curve.

2.3. Check for the correlation of the variables. Using heatmap we can find out the correlation of variables:

	age	economic.cond.national \
age	1.000000	0.018687
economic.cond.national	0.018687	1.000000
economic.cond.household	-0.038868	0.347687
Blair	0.032084	0.326141
Hague	0.031144	-0.200790
Europe	0.064562	-0.209150
political.knowledge	-0.046598	-0.023510

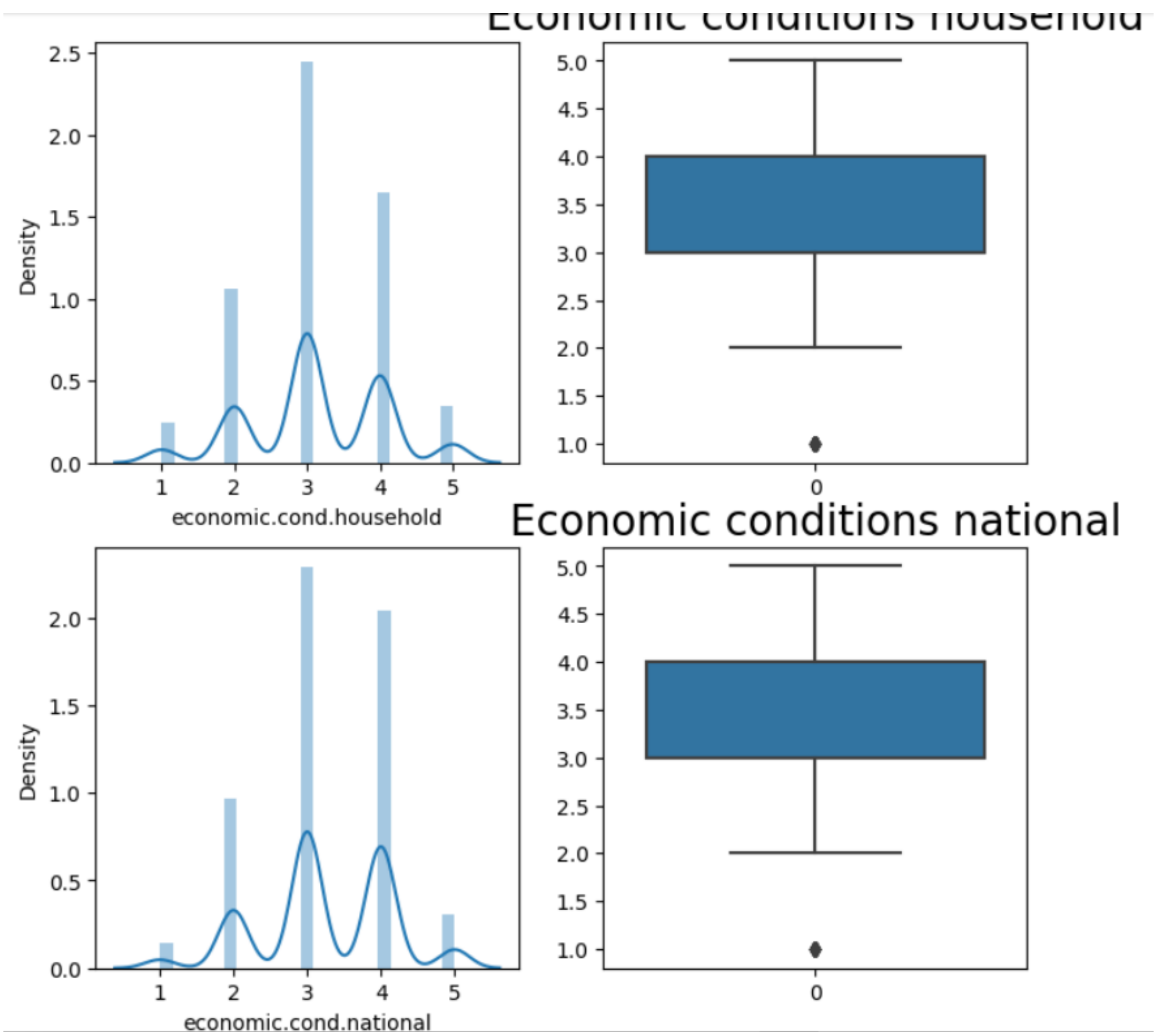
	economic.cond.household	Blair	Hague \
age	-0.038868	0.032084	0.031144
economic.cond.national	0.347687	0.326141	-0.200790
economic.cond.household	1.000000	0.215822	-0.100392
Blair	0.215822	1.000000	-0.243508
Hague	-0.100392	-0.243508	1.000000
Europe	-0.112897	-0.295944	0.285738
political.knowledge	-0.038528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

	Europe	political.knowledge
age	0.064562	-0.046598
economic.cond.national	-0.209150	-0.023510
economic.cond.household	-0.112897	-0.038528
Blair	-0.295944	-0.021299
Hague	0.285738	-0.029906
Europe	1.000000	-0.151197
political.knowledge	-0.151197	1.000000

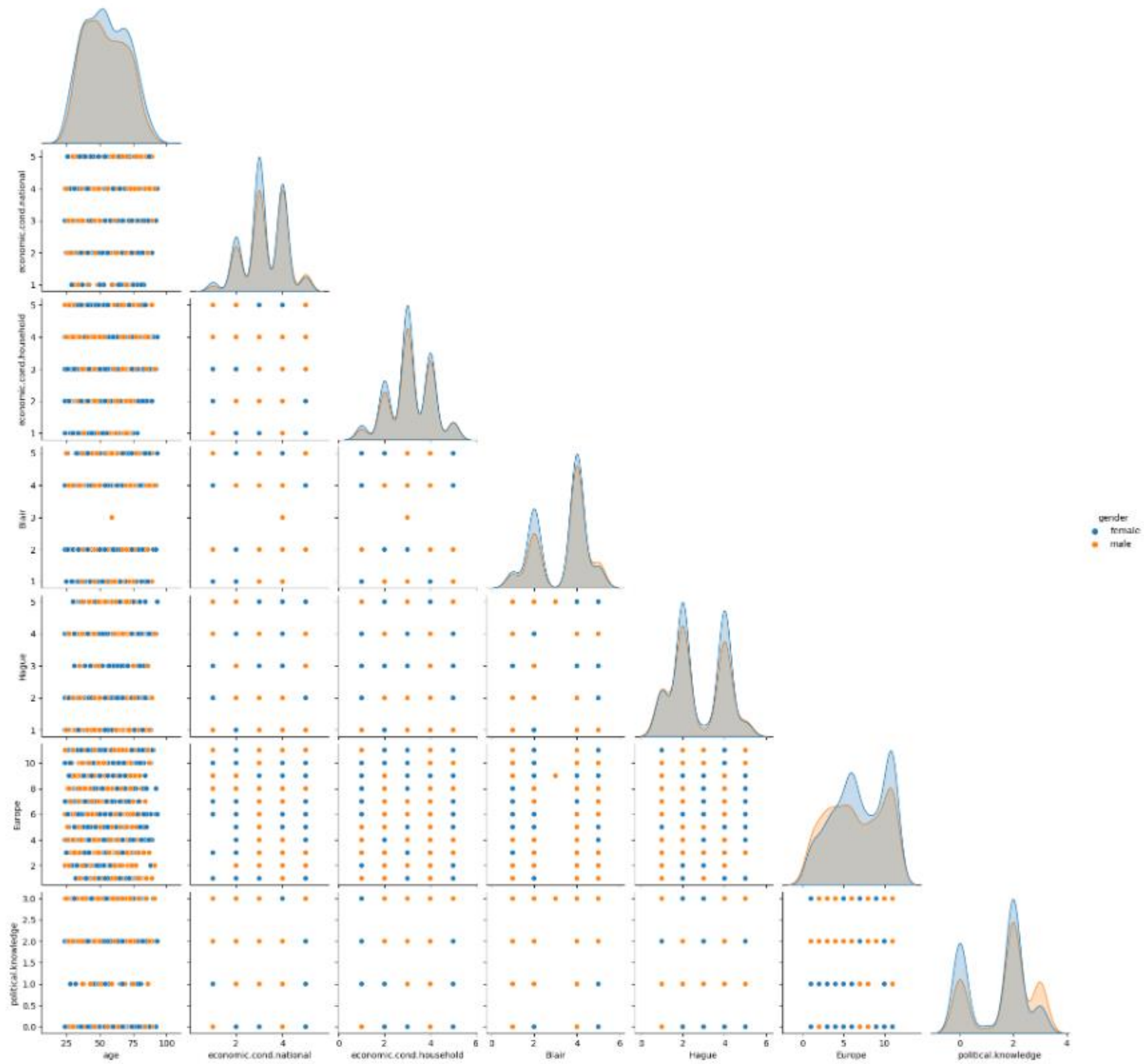


Most of the variables are negatively correlated and there is less chance of multicollinearity problem in the variables.

2.4. Checking the distribution of variables with outliers in detail by plotting histogram and boxplot simultaneously.



2.5. Segregate all the variables using their gender distribution:



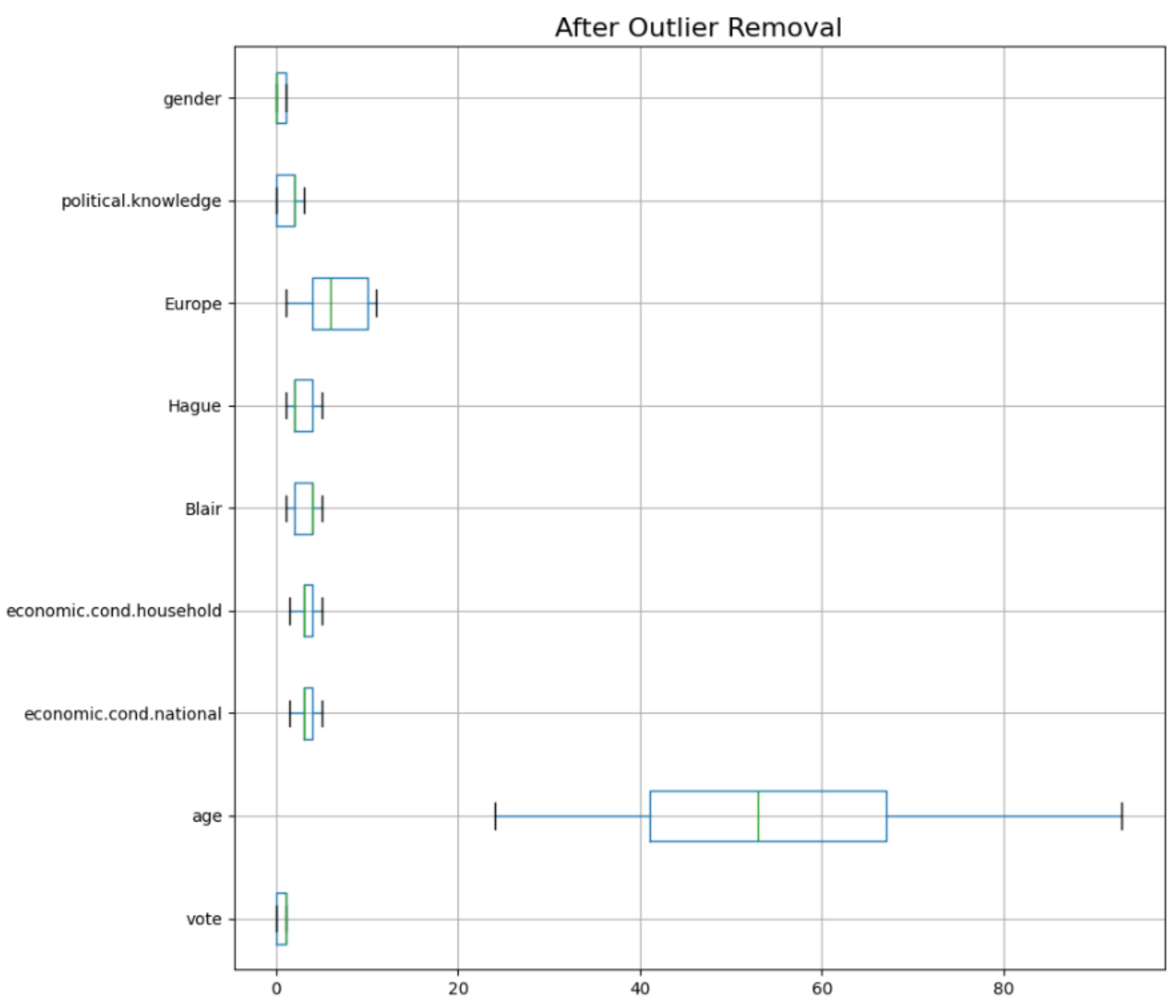
Here, Orange represents the male distribution and blue represents the female distribution. Most of the observations are from male gender as compared to female.

2.6. Check for any imbalance problem in the dataset. Let us use the vote variable to find out the number of labour and conservative counts in the dataset.

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

Here, most of the people have voted for labour parites and 30% only have voted for conservatives.

2.7. Remove the outliers from the dataset by performing IQR method. Using boxplot will show outlier treatment.



2.8. Using vote variable as our target variable and to find out the performance metrics for both the labour and conservative parties. It is visible that most of the people have voted for labour parties and 30% only have voted for conservative. To predict the voters using this target variables. For predicting the model split the dataset into training and testing dataset.Also converting the categorical variable into one hot encoding . Also convert the object datatypes into numerical values for modelling.

```
VOTE : 2
vote
0     460
1    1057
Name: count, dtype: int64


GENDER : 2
gender
1    709
0    808
Name: count, dtype: int64
```

So vote has two codes 0 represents conservative and 1 represents labour votes.
Gender variable also assigned with respective codes 0s and 1s.

2.9. Checking the dataset is ready for split into training and testing:

```
<class 'pandas.core.frame.DataFrame'>
Index: 1517 entries, 0 to 1524
Data columns (total 9 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   vote                     1517 non-null   float64
 1   age                      1517 non-null   float64
 2   economic.cond.national   1517 non-null   float64
 3   economic.cond.household  1517 non-null   float64
 4   Blair                    1517 non-null   float64
 5   Hague                    1517 non-null   float64
 6   Europe                   1517 non-null   float64
 7   political.knowledge      1517 non-null   float64
 8   gender                   1517 non-null   float64
dtypes: float64(9)
memory usage: 150.8 KB
```

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	1.0	43.0	3.0	3.0	4.0	1.0	2.0	2.0	0.0
1	1.0	36.0	4.0	4.0	4.0	4.0	5.0	2.0	1.0
2	1.0	35.0	4.0	4.0	5.0	2.0	3.0	2.0	1.0
3	1.0	24.0	4.0	2.0	2.0	1.0	4.0	0.0	0.0
4	1.0	41.0	2.0	2.0	1.0	1.0	6.0	2.0	1.0

2.10. Perform Scaling only for KNN modeling. As scaling is necessary only for model which is based on distance rule. LDA and Logistic and Naives bayes doesn’t have any effects by scaling.
So, split the dataset as 70% for training and 30% for test. As vote is target variable and others are independent variables.
Here are training dataset observations:

```
age                        68
economic.cond.national      5
economic.cond.household     5
Blair                       5
Hague                       5
Europe                     11
political.knowledge         4
gender                      2
dtype: int64
```

Here are testing dataset:

```
age                          66
economic.cond.national        5
economic.cond.household       5
Blair                         4
Hague                         5
Europe                       11
political.knowledge           4
gender                        2
dtype: int64
```

2.11. Size of training and testing datasets:

```
(1061, 8)
(456, 8)
```

Question.3. Model Building

3.1. For model building importing logistic regression from sklearn_library. Let us call the function into a variable called model with a limit of max iteration of 1000. Let us call the model from library and assign to a new variable:

```
▾ LogisticRegression
LogisticRegression(max_iter=1000)
```

```
▾ LinearDiscriminantAnalysis
LinearDiscriminantAnalysis()
```

```
▾ GaussianNB
GaussianNB()
```

3.2. After calling the model function from library, Fit the training dataset into the respective model variables and check for the model accuracy for both training and testing dataset.

```
The LDA training accuracy : 0.8397737983034873
The LDA testing accuracy: 0.8223684210526315
```

```
The logistic reg training model accuracy : 0.8454288407163054
The logistic reg testing model accuracy : 0.8223684210526315
```

Here, it is visible that there is small difference in training accuracies. Both models has same testing accuracy. Also both does not require scaling because both are not affected by scaled values.

3.3. Here importing naive bayes and KNN model from sklearn library.
For KNN model scale the dataset and split them into training and testing.
For naive bayes no scaling the data.
For checking the prediction fit the training dataset into model.

	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	-0.716161	-0.301648	-0.179682	0.565802	-1.419969	-1.437338	0.423832	-0.936736
1	-1.162118	0.870183	0.949003	0.565802	1.014951	-0.527684	0.423832	1.067536
2	-1.225827	0.870183	0.949003	1.417312	-0.608329	-1.134120	0.423832	1.067536
3	-1.926617	0.870183	-1.308366	-1.137217	-1.419969	-0.830902	-1.421084	-0.936736
4	-0.843577	-1.473479	-1.308366	-1.988727	-1.419969	-0.224465	0.423832	1.067536
...	...	...	...	...	...	...	...	...
1512	0.812836	2.042014	-0.179682	-1.137217	1.014951	1.291625	1.346290	1.067536
1513	1.195085	-1.473479	-1.308366	0.565802	1.014951	0.381971	0.423832	1.067536
1514	-1.098410	-0.301648	-0.179682	1.417312	1.014951	-1.437338	0.423832	1.067536
1515	0.430587	-0.301648	-0.179682	-1.988727	1.014951	1.291625	0.423832	1.067536
1516	1.258794	-1.473479	-0.179682	-1.137217	1.014951	1.291625	-1.421084	-0.936736

```
The Naive Bayes training accuracy:  0.8435438265786993
The Naive Bayes testing accuracy: 0.8245614035087719
```

```
The KNN training accuracy : 0.8708765315739868
The KNN testing accuracy: 0.8223684210526315
```

From these model accuracies, KNN model have more accuracy on training dataset compares to all other model. All model have overfit problems. Let us consider models with which less than 10% difference in training and testing as a standard model. So, let us perform prediction with all models and heck their precision, recall, f1-score and model accuracy. We will be considering the F1-score to compare the model efficiency.

3.4. Importing GridSearchCV from sklearn.model_selection for hypertuning the models. Before applying the bagging classifier, Let us use random forest and apply into bagging base estimator. Let us apply gradiest boosting and adaboost classifier as boosting alogirthm . Naive bayes doesn't hold with more number of parameters. So it is difficult to tune the naive bayes alogorithm. We will be tuning the all the base models with different combination of parameters.

3.5. Logistic Regression TUNED:

After applying the gridsearchCv function, we got the best estimator from the model with all permutation and combination of parameters within models.

```
{'penalty': 'l2', 'random_state': 0, 'solver': 'newton-cg', 'tol': 0.0001}
```

3.6. LDA TUNED:

After applying the different combination of parameters into the gridsearchCv function, we got as below the best parameter for LDA

```
{'solver': 'lsqr', 'tol': 0.0001}
```

3.7. KNN TUNED:

With 9 combination of cross validation , we got below parameters as the best parameter for knn model.

```
{'algorithm': 'auto', 'leaf_size': 30, 'p': 1, 'weights': 'uniform'}
```

We will go with the base model parameter for naives bayes. Since naïve bayes doesn't have more parameters to tune.

3.8. BAGGING:

For Bagging, let us import RandomForestClassifier from sklearnmodel library.

Apply all the parameters and fit the model.Before that let us find out out of bag score error to identify the amount of error in the prediction.

```
0.824693685202639
0.824693685202639
0.820923656927427
0.8265786993402451
0.8369462770970783
0.827521206409048
0.8303487276154571
0.8341187558906692
0.8294062205466541
0.8294062205466541
0.8331762488218661
0.8162111215834119
```

Above are the output of OOB score for respective random state, lets us choose random state as 5 to have a better prediction from the above OOB score. After applying the parameter into randomforeset model. Let us fit the training variables into the model and do the bagging operation. RFC used as the base estimator in the bagging classifer model.

```
RandomForestClassifier
RandomForestClassifier(n_estimators=501, random_state=5)
```

3.9. BOOSTING:

From sklearn.ensemble importing AdaBoostClassifier also, importing Gradientboostingclassifier:

abcl = AdaBoostClassifier(n_estimators=10, random_state=1)
gbcl = GradientBoostingClassifier(n_estimators = 50,random_state=1)

3.10. Performance Metrics:

Let us evaluate the training and testing accuracy , precision, recall, f1-score respectively.

Logistic Regression:

Logistic Regression - Training:

```
[[218 103]
 [ 61 679]]
```

	precision	recall	f1-score	support
0.0	0.78	0.68	0.73	321
1.0	0.87	0.92	0.89	740
accuracy			0.85	1061
macro avg	0.82	0.80	0.81	1061
weighted avg	0.84	0.85	0.84	1061

Logistic Regression Tuned - Training:

```
array([[218, 103],
       [ 61, 679]], dtype=int64)
```

	precision	recall	f1-score	support
0.0	0.78	0.68	0.73	321
1.0	0.87	0.92	0.89	740
accuracy			0.85	1061
macro avg	0.82	0.80	0.81	1061
weighted avg	0.84	0.85	0.84	1061

Logistic Regression - Testing

```
[[ 92  47]
 [ 34 283]]
```

	precision	recall	f1-score	support
0.0	0.73	0.66	0.69	139
1.0	0.86	0.89	0.87	317
accuracy			0.82	456
macro avg	0.79	0.78	0.78	456
weighted avg	0.82	0.82	0.82	456

Logistic Regression Tuned testing :

```
array([[ 92,  47],
       [ 34, 283]], dtype=int64)
```

	precision	recall	f1-score	support
0.0	0.73	0.66	0.69	139
1.0	0.86	0.89	0.87	317
accuracy			0.82	456
macro avg	0.79	0.78	0.78	456
weighted avg	0.82	0.82	0.82	456

There is 87% of f1-score on labour votes and 69% of prediction for conservative voters.

Linear Discriminant Analysis:

LDA Training :

```
[[221 100]
 [ 70 670]]
```

	precision	recall	f1-score	support
0.0	0.76	0.69	0.72	321
1.0	0.87	0.91	0.89	740
accuracy			0.84	1061
macro avg	0.81	0.80	0.80	1061
weighted avg	0.84	0.84	0.84	1061

LDA Training Tuned:

```
array([[221, 100],
       [ 70, 670]], dtype=int64)
```

	precision	recall	f1-score	support
0.0	0.76	0.69	0.72	321
1.0	0.87	0.91	0.89	740
accuracy			0.84	1061
macro avg	0.81	0.80	0.80	1061
weighted avg	0.84	0.84	0.84	1061

LDA Testing:

```
[[ 94  45]
 [ 36 281]]
```

	precision	recall	f1-score	support
0.0	0.72	0.68	0.70	139
1.0	0.86	0.89	0.87	317
accuracy			0.82	456
macro avg	0.79	0.78	0.79	456
weighted avg	0.82	0.82	0.82	456

LDA Testing tuned:

```
array([[ 94,  45],
       [ 36, 281]], dtype=int64)
```

	precision	recall	f1-score	support
0.0	0.72	0.68	0.70	139
1.0	0.86	0.89	0.87	317
accuracy			0.82	456
macro avg	0.79	0.78	0.79	456
weighted avg	0.82	0.82	0.82	456

There is no difference between base model and tuned model. 87% prediction is for labour voters and 70% for f1-score is conservative voters are correctly and more accurately predicted in LDA model to Logistic regression model.

Naive Bayes Model:

Naive Bayes Model - Training:

```
[[233  88]
 [ 78 662]]
```

	precision	recall	f1-score	support
0.0	0.75	0.73	0.74	321
1.0	0.88	0.89	0.89	740
accuracy			0.84	1061
macro avg	0.82	0.81	0.81	1061
weighted avg	0.84	0.84	0.84	1061

Naive Bayes Model – Testing:

```
[[ 94  45]
 [ 36 281]]
```

	precision	recall	f1-score	support
0.0	0.72	0.68	0.70	139
1.0	0.86	0.89	0.87	317
accuracy			0.82	456
macro avg	0.79	0.78	0.79	456
weighted avg	0.82	0.82	0.82	456

There is 1% increase in prediction for labour voters with naive bayes model while 70% of f1-score for conservative voters.

KNN Model

KNN Model – Training:

```
0.8708765315739868
```

```
[[238  83]
 [ 54 686]]
```

	precision	recall	f1-score	support
0.0	0.82	0.74	0.78	321
1.0	0.89	0.93	0.91	740
accuracy			0.87	1061
macro avg	0.85	0.83	0.84	1061
weighted avg	0.87	0.87	0.87	1061

KNN Model Tuned Training:

```
array([[242,  79],
       [ 53, 687]], dtype=int64)
```

	precision	recall	f1-score	support
0.0	0.82	0.75	0.79	321
1.0	0.90	0.93	0.91	740
accuracy			0.88	1061
macro avg	0.86	0.84	0.85	1061
weighted avg	0.87	0.88	0.87	1061

After tune model accuracy has been increased.

KNN Model Testing:

```
0.8223684210526315
```

```
[[ 94  45]
 [ 36 281]]
```

	precision	recall	f1-score	support
0.0	0.72	0.68	0.70	139
1.0	0.86	0.89	0.87	317
accuracy			0.82	456
macro avg	0.79	0.78	0.79	456
weighted avg	0.82	0.82	0.82	456

KNN Model Tuned Testing:

```
array([[ 94,  45],
       [ 35, 282]], dtype=int64)
```

	precision	recall	f1-score	support
0.0	0.73	0.68	0.70	139
1.0	0.86	0.89	0.88	317
accuracy			0.82	456
macro avg	0.80	0.78	0.79	456
weighted avg	0.82	0.82	0.82	456

It is visible that after tuning the f1-score for labour voters have increased by 1% on testing report.

BAGGING:

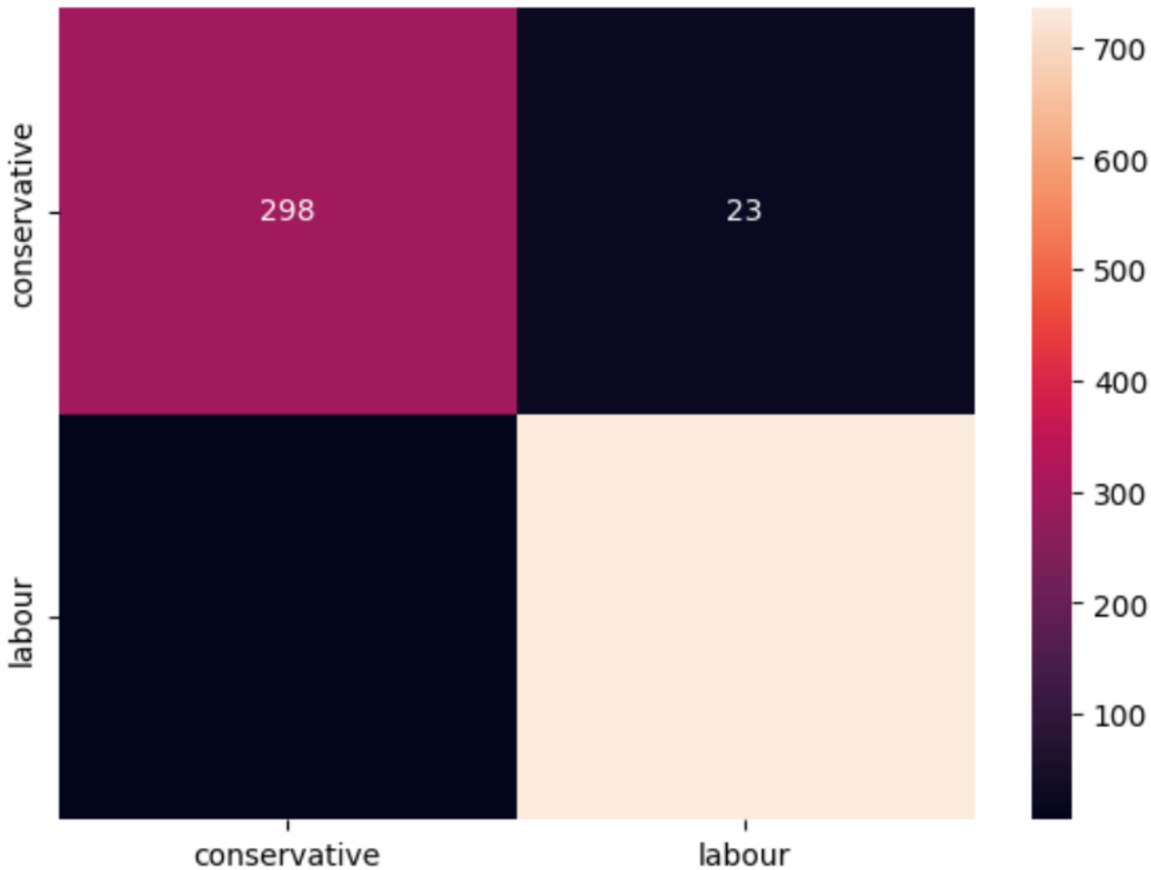
Importing RandomForestClassifier and fitted the training dataset into it. Using the RandomForestClassifier as the base estimator for the bagging classifier.

Bagging Training:
Confusion matrix:

```
0.9736098020735156
```

[172]:

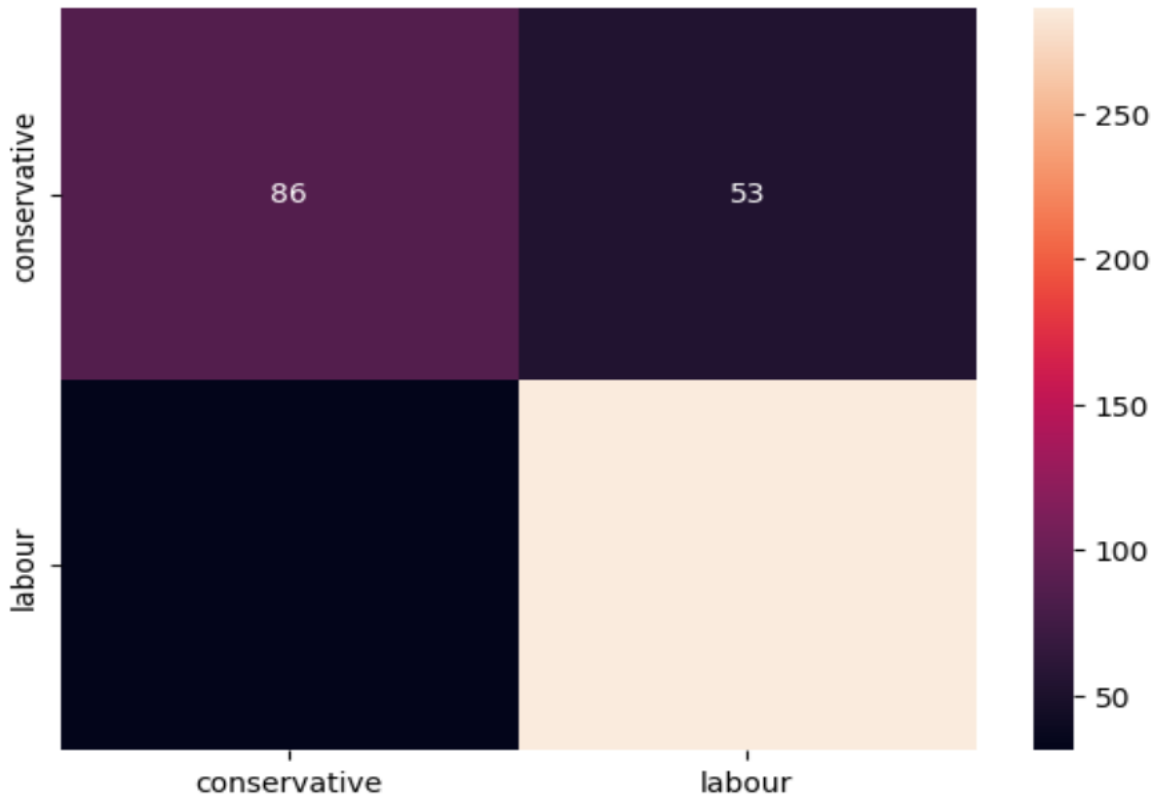
<Axes: >



	precision	recall	f1-score	support
0.0	0.98	0.93	0.96	321
1.0	0.97	0.99	0.98	740
accuracy			0.97	1061
macro avg	0.98	0.96	0.97	1061
weighted avg	0.97	0.97	0.97	1061

Bagging Testing:

```
0.8157894736842105
[175]:
<Axes: >
```



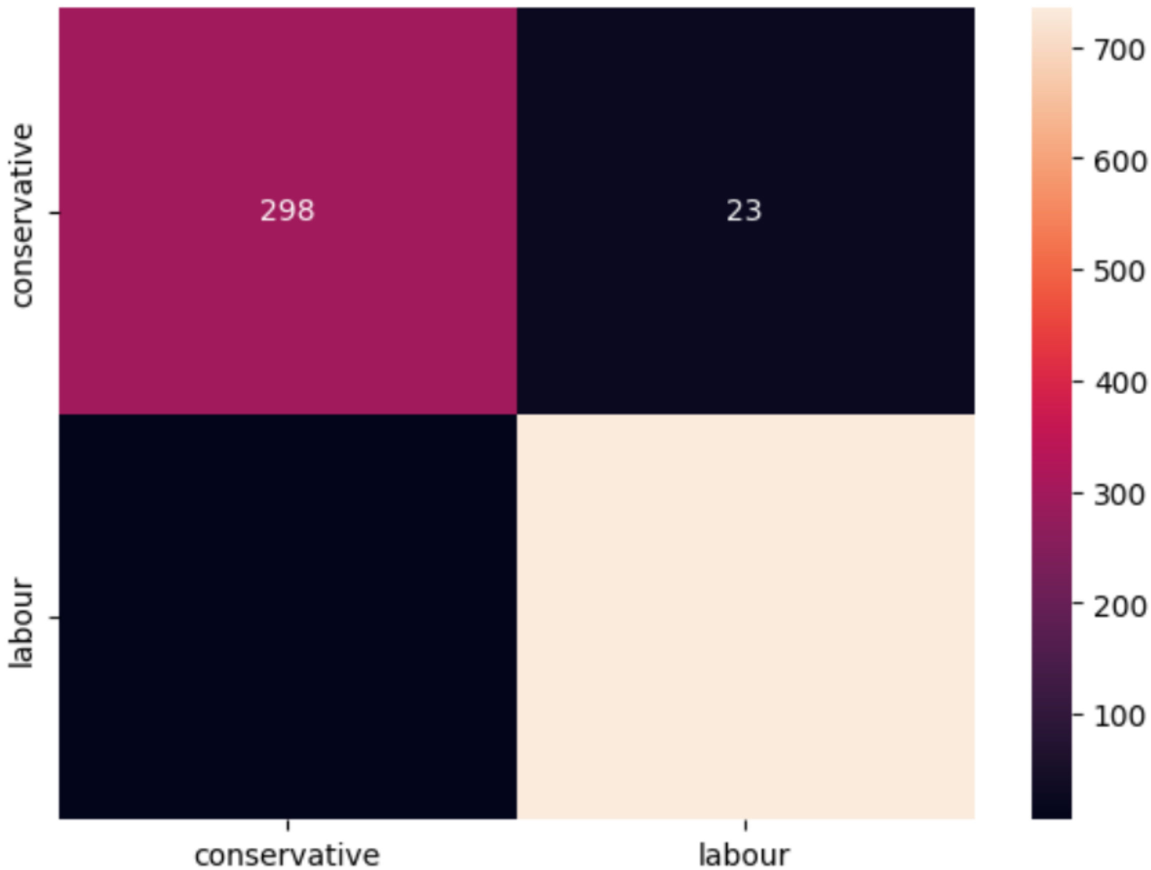

```
array([[ 86,  53],
       [ 31, 286]], dtype=int64)
```

	precision	recall	f1-score	support
0.0	0.74	0.62	0.67	139
1.0	0.84	0.90	0.87	317
accuracy			0.82	456
macro avg	0.79	0.76	0.77	456
weighted avg	0.81	0.82	0.81	456

After applying bagging classifier we received 97% of nodel accuracy. After checking performance of testing model we received 82% of model accuracy. Hence there is 15% difference in training and testing model accuracies of bagging classifier.

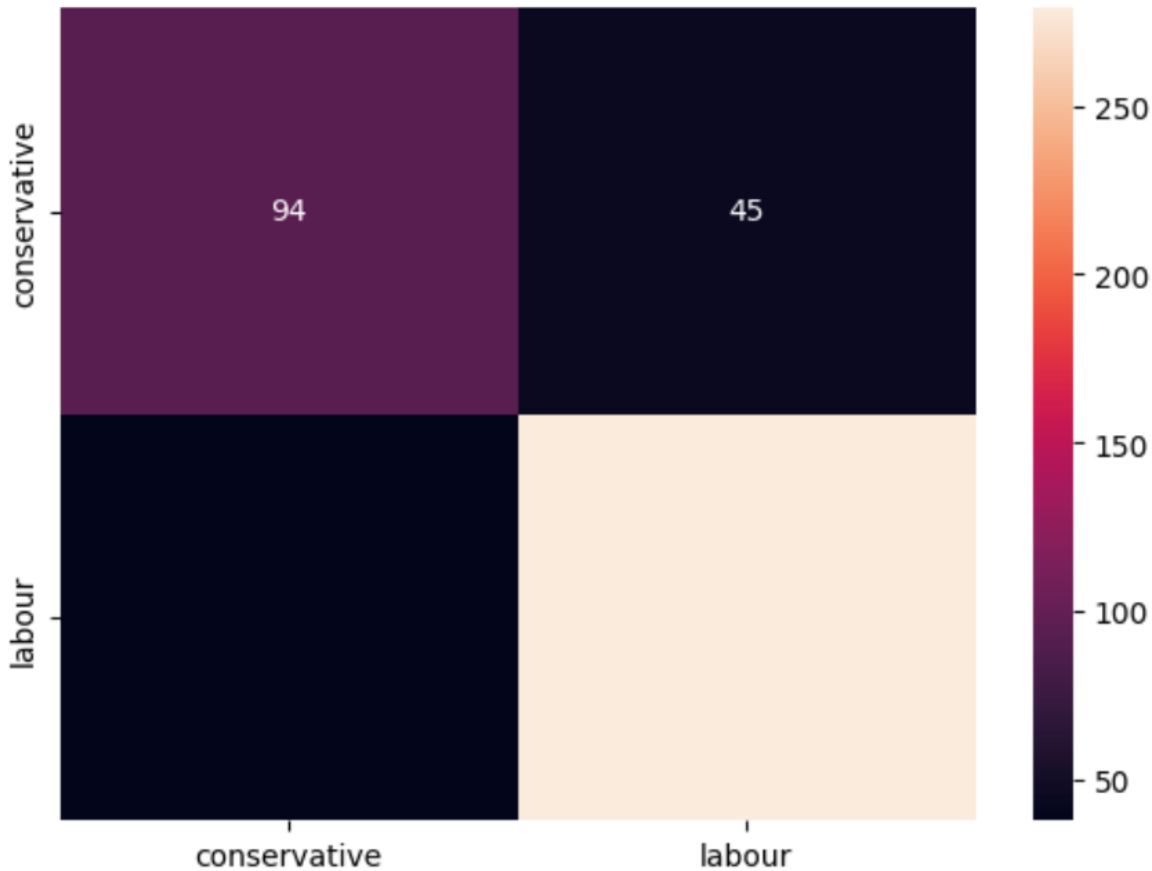
BOOSTING:

```
0.9736098020735156
[172]:
<Axes: >
```



	precision	recall	f1-score	support
0.0	0.98	0.93	0.96	321
1.0	0.97	0.99	0.98	740
accuracy			0.97	1061
macro avg	0.98	0.96	0.97	1061
weighted avg	0.97	0.97	0.97	1061

```
0.8179824561403509
[183]:

<Axes: >
```



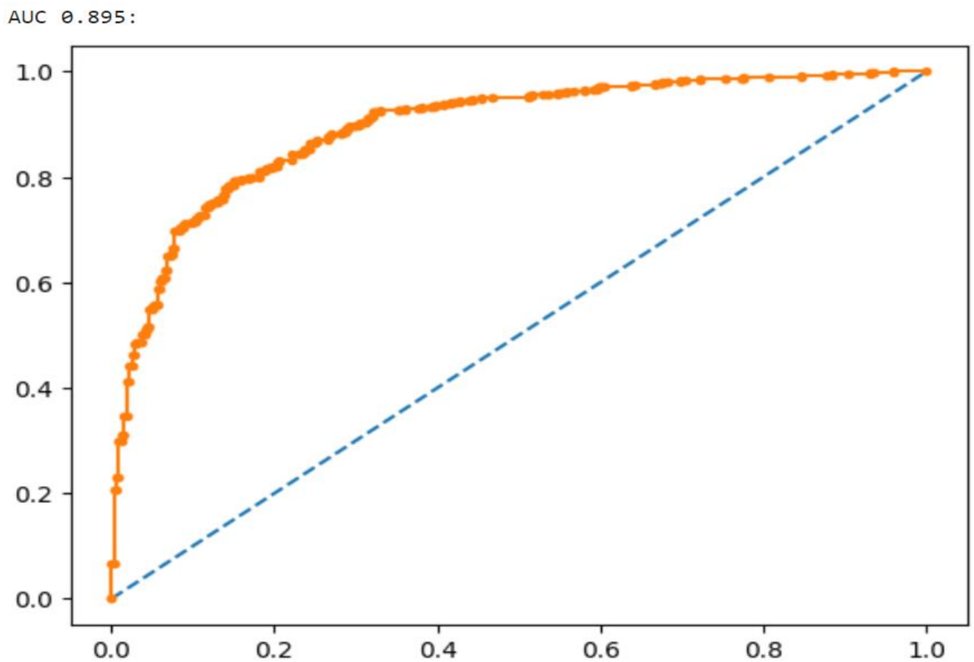

```
              precision    recall  f1-score   support

         0.0       0.71      0.68      0.69       139
         1.0       0.86      0.88      0.87       317

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

Here both bagging and boosting models have overfitting problems. Both the model have same testing accuracy.
It is visible that , boosting f1-score for conservative voters 2% higher than to bagging. Hence, boosting is better compared to bagging.

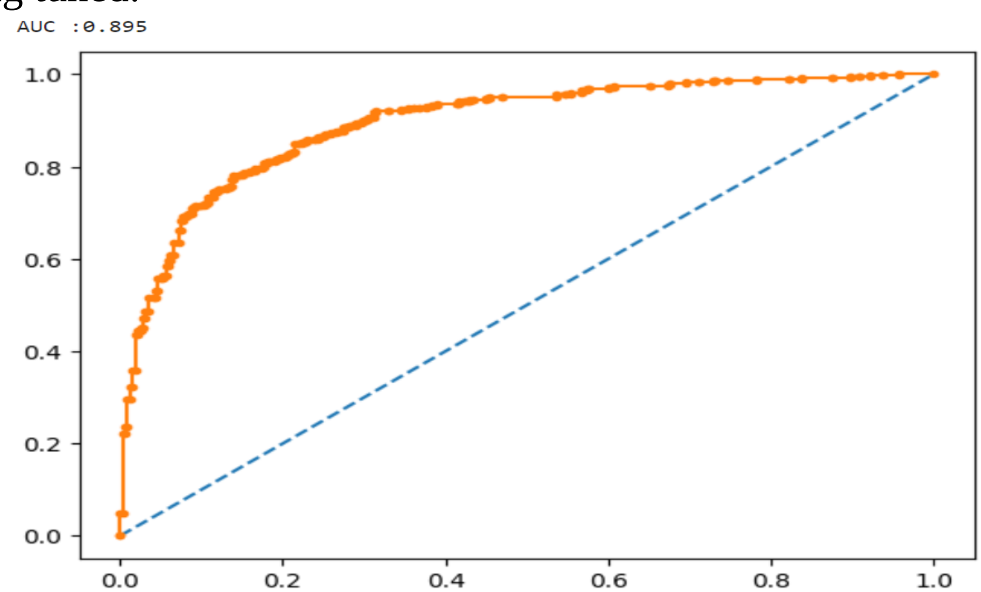
Question 4. Model Performance evaluation:
ROC AND AUC CURVE:

4.1. Logistic Regression:

Training:

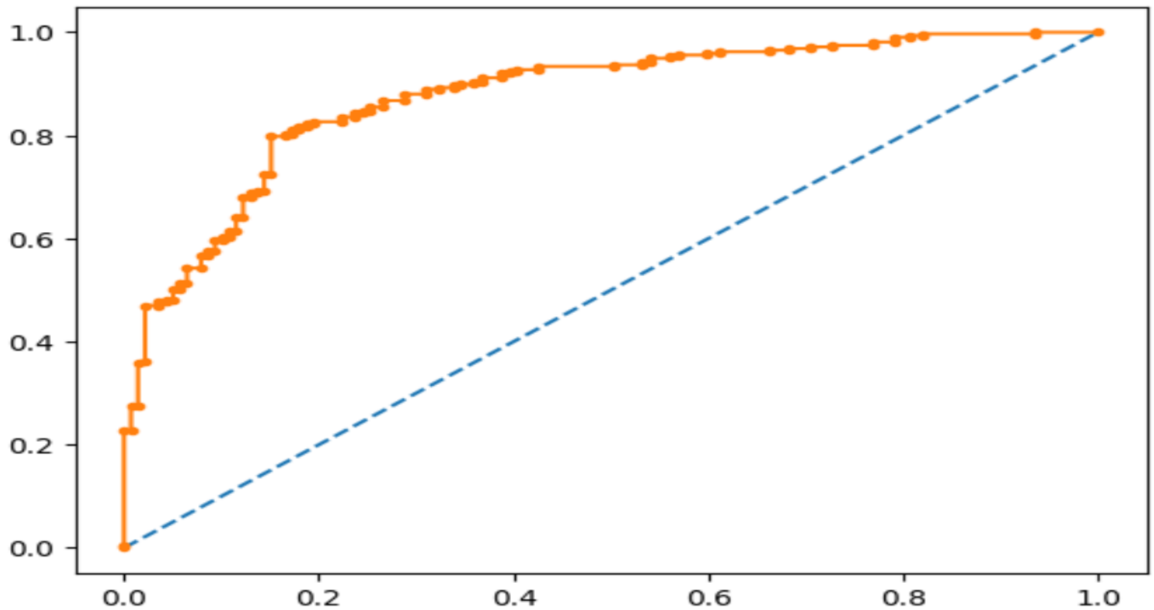


Training tuned:



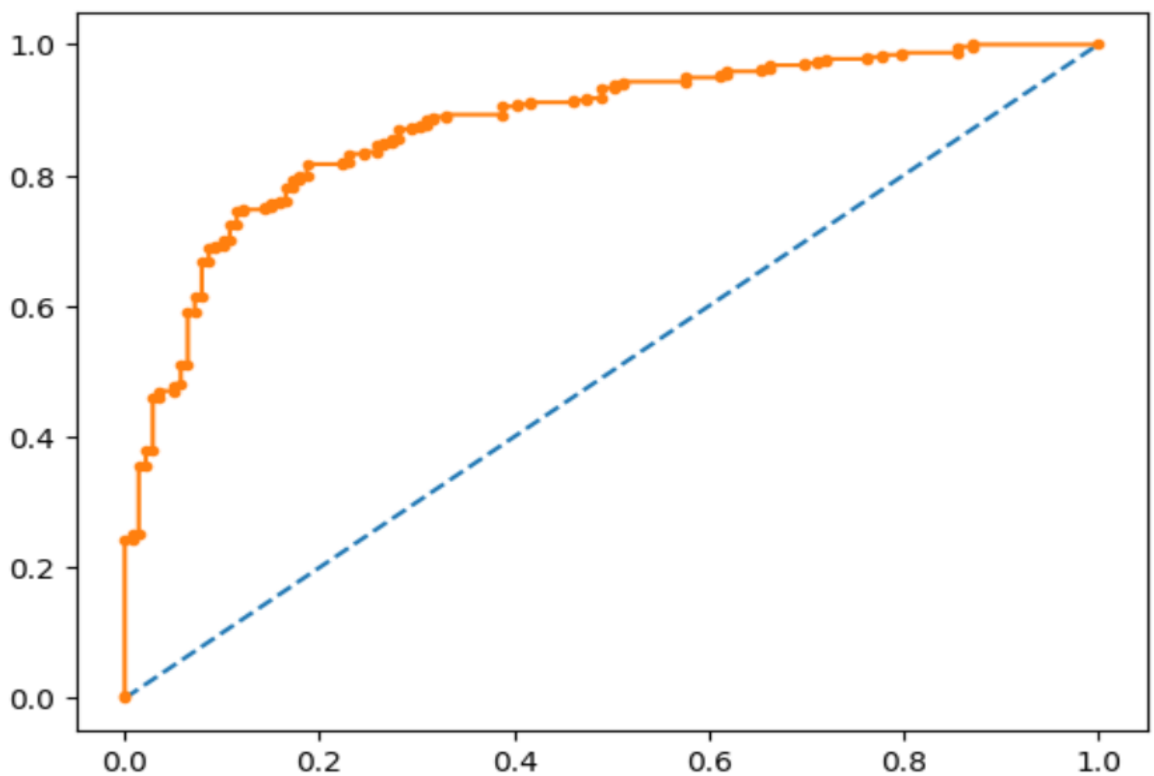
Testing:

AUC Score : 0.877



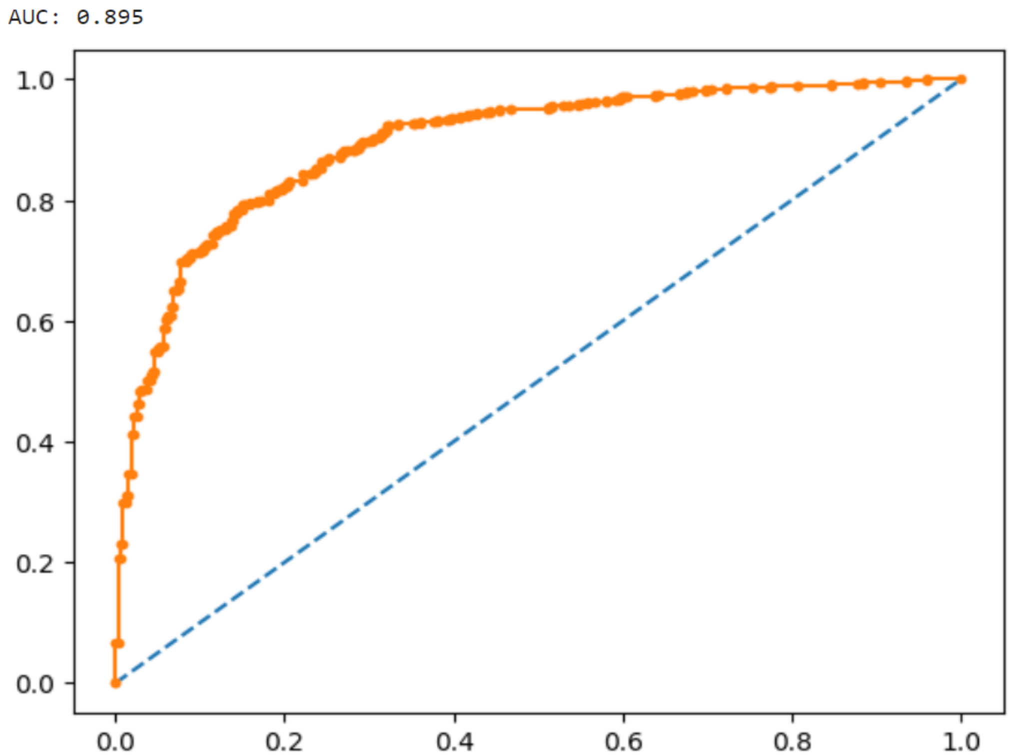
Testing Tuned:

AUC :0.877

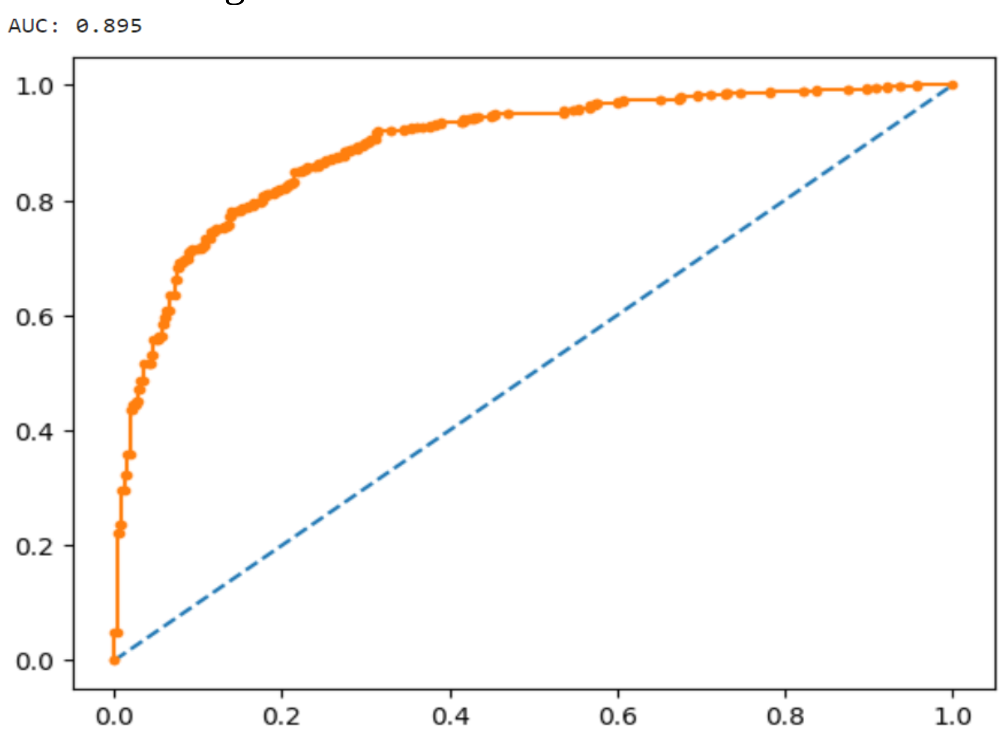


4.2. LDA

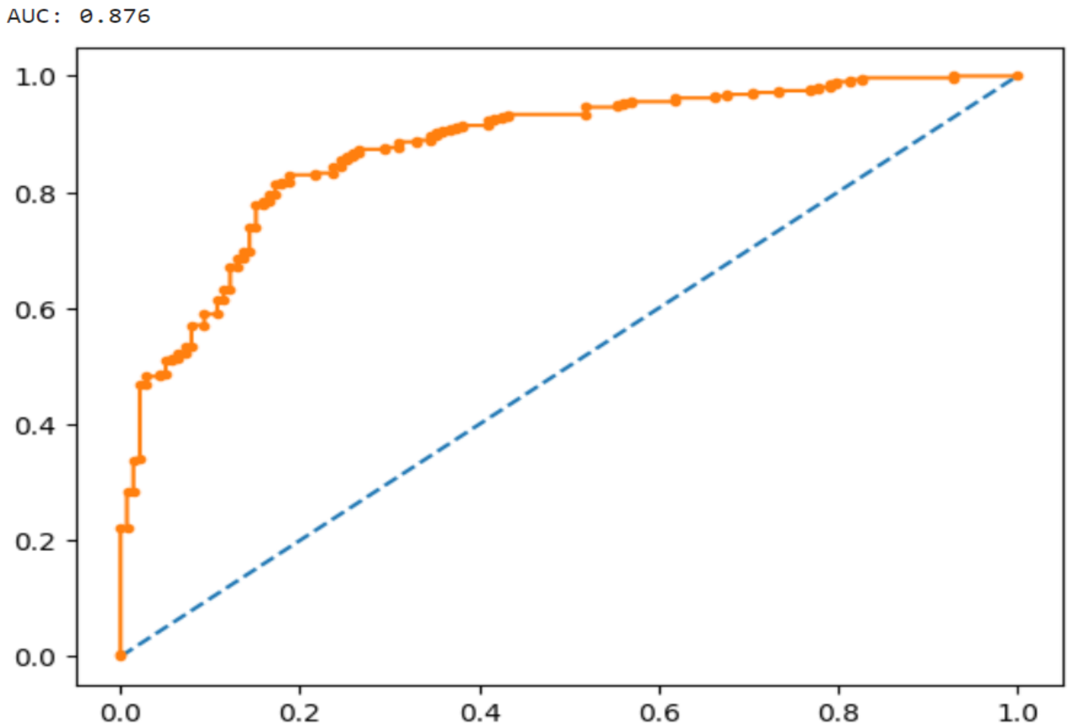
LDA Training:



LDA Tuned Training:

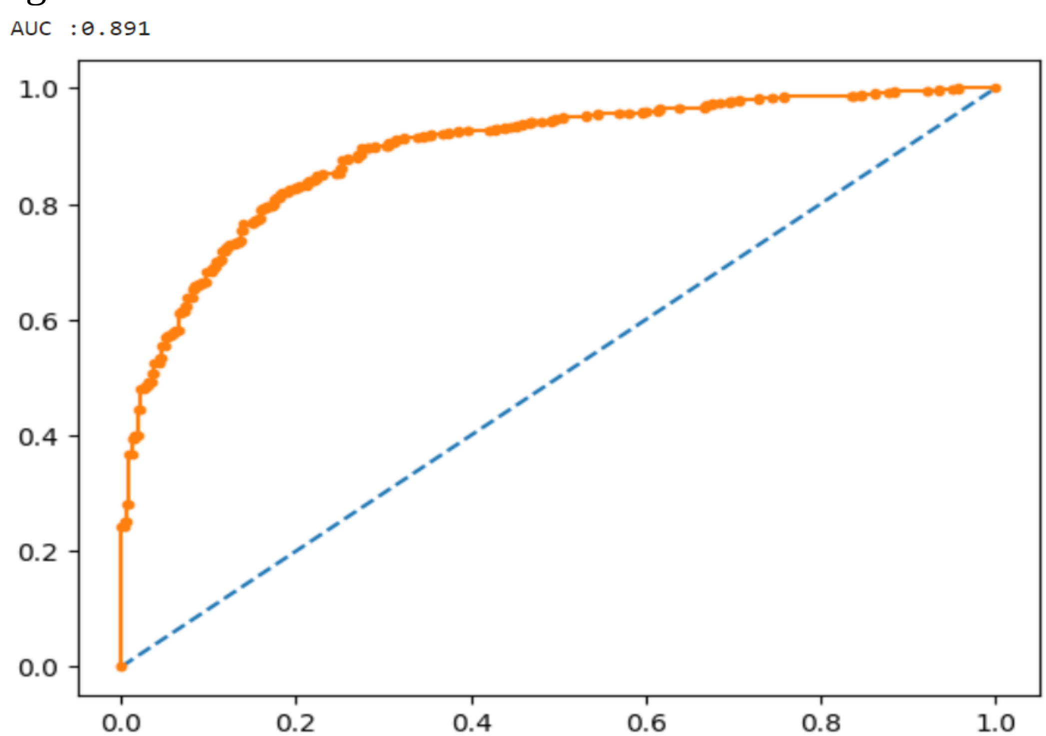


LDA Testing:



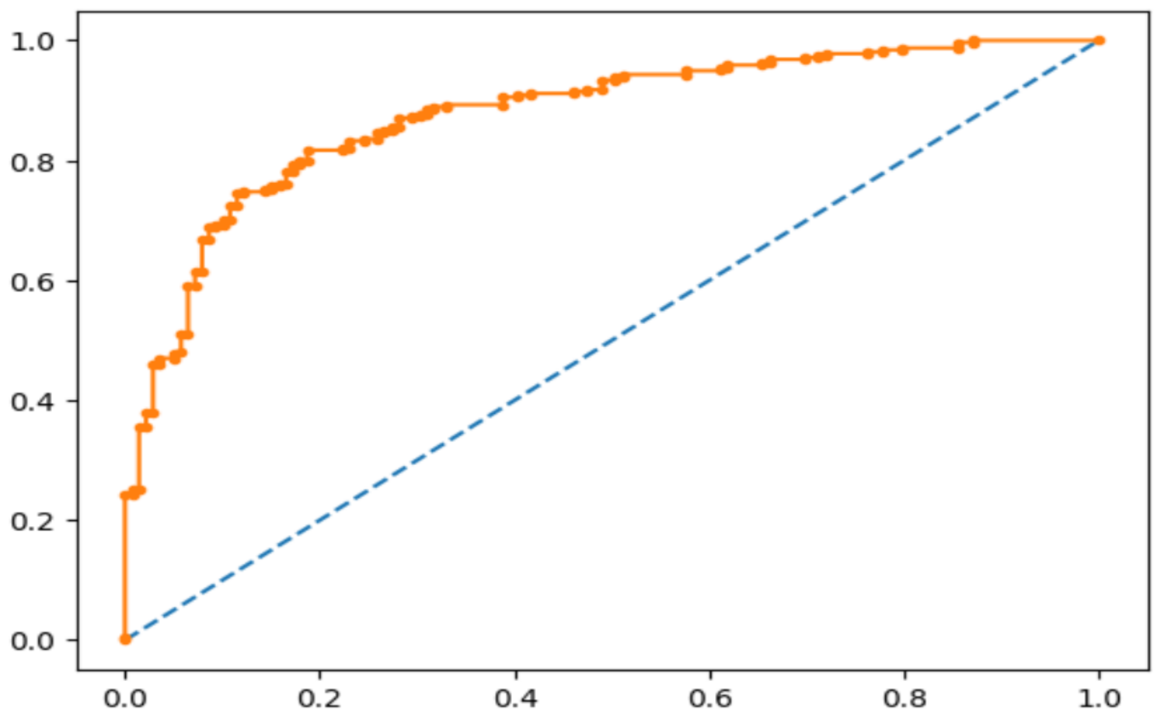
Naive Bayes:

Training:



Testing:

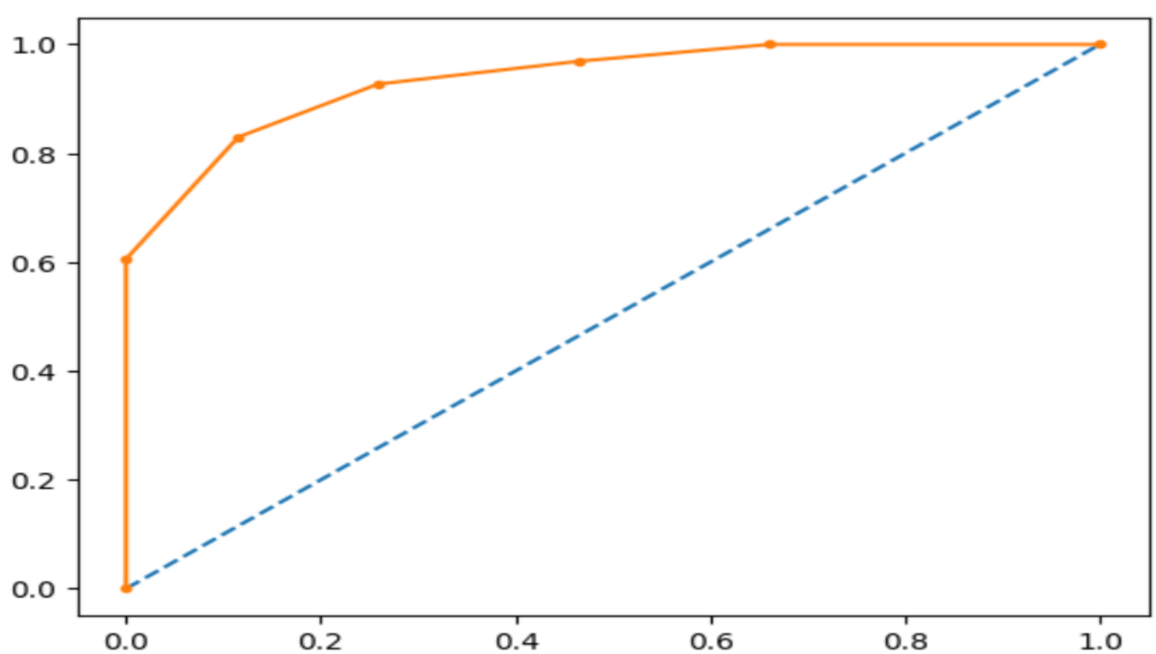
AUC :0.877



4.4 KNN

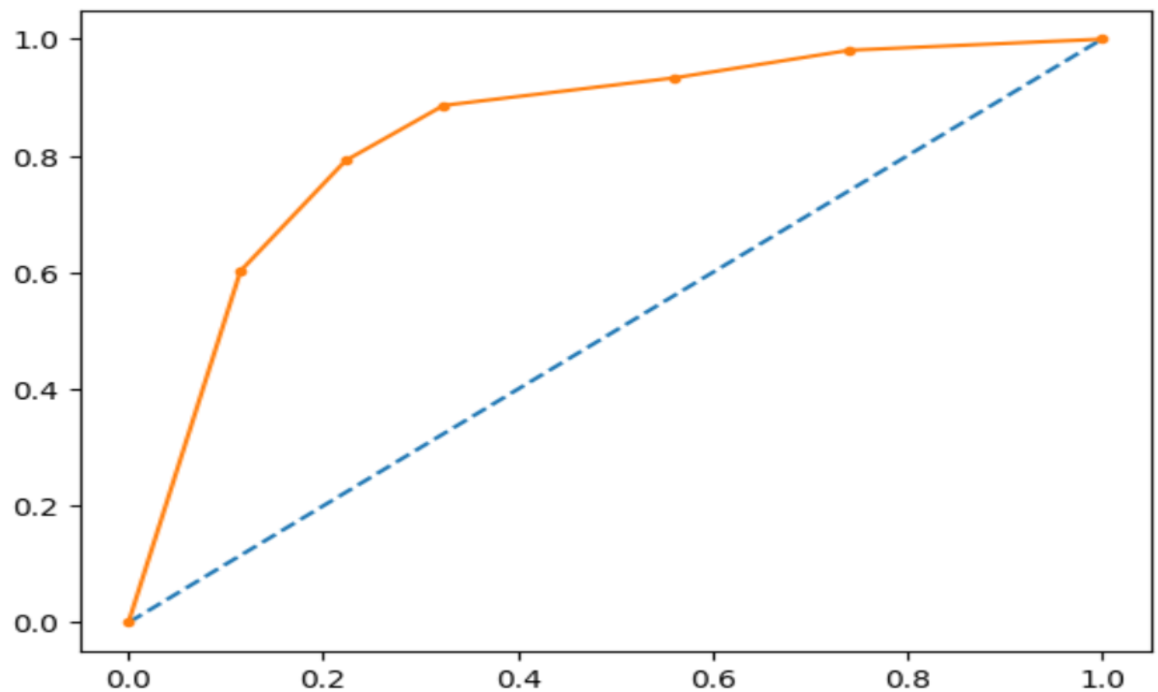
KNN Training:

AUC: 0.936



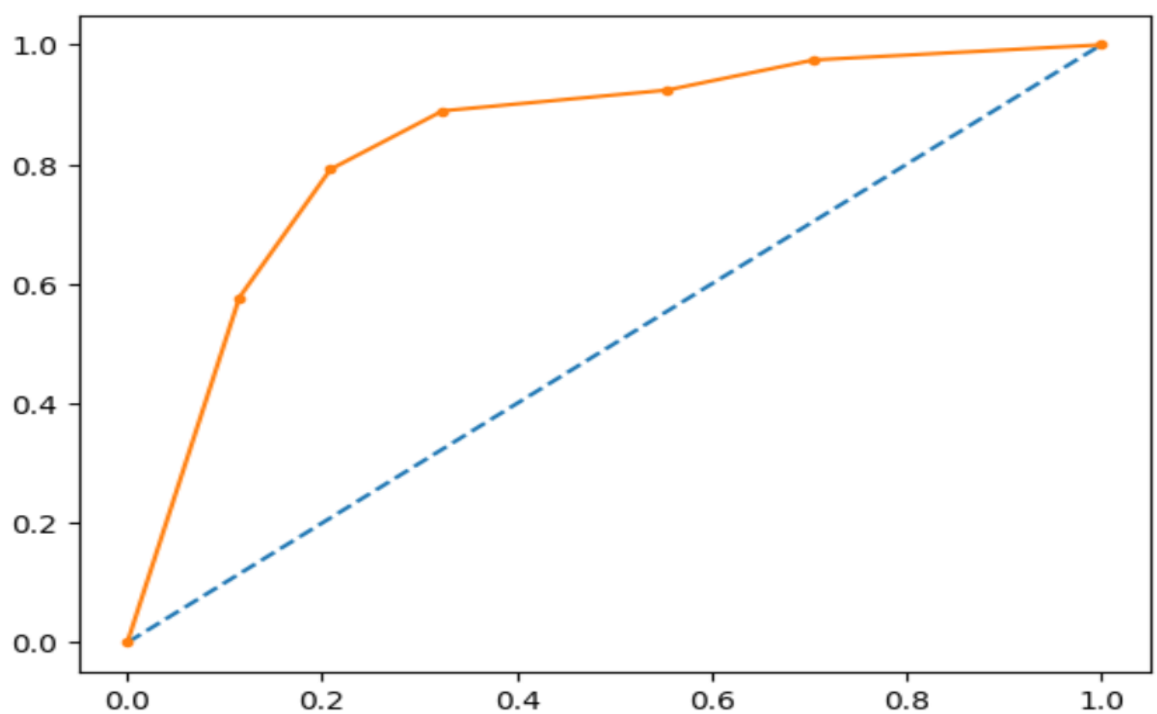
KNN Tuned Training:

AUC: 0.839



KNN Tuned Testing:

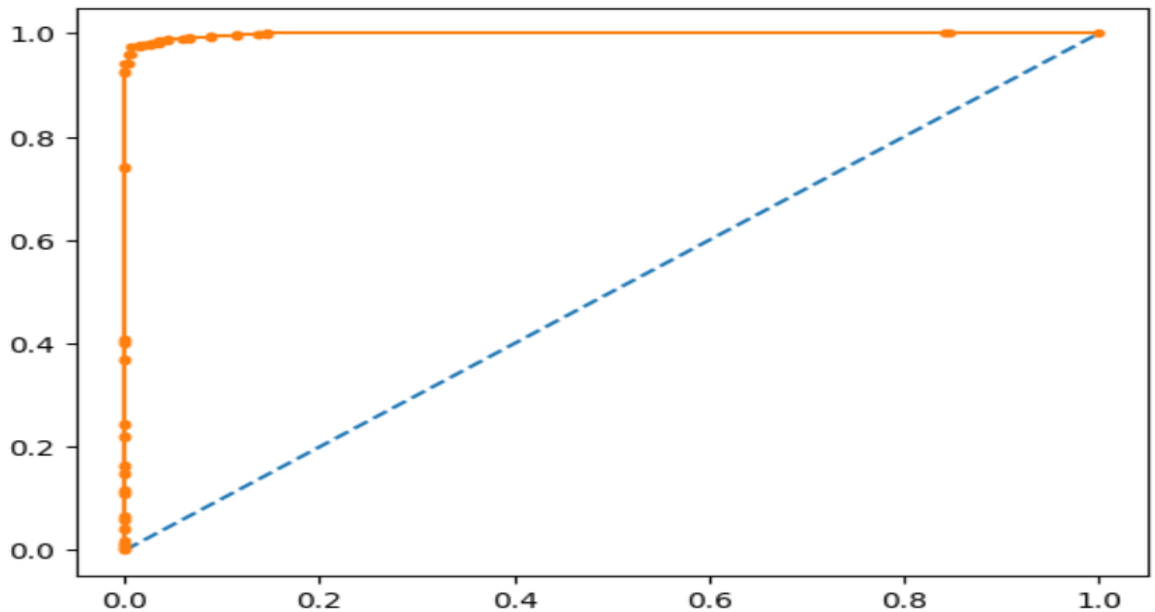
AUC: 0.838



4.5. Bagging:

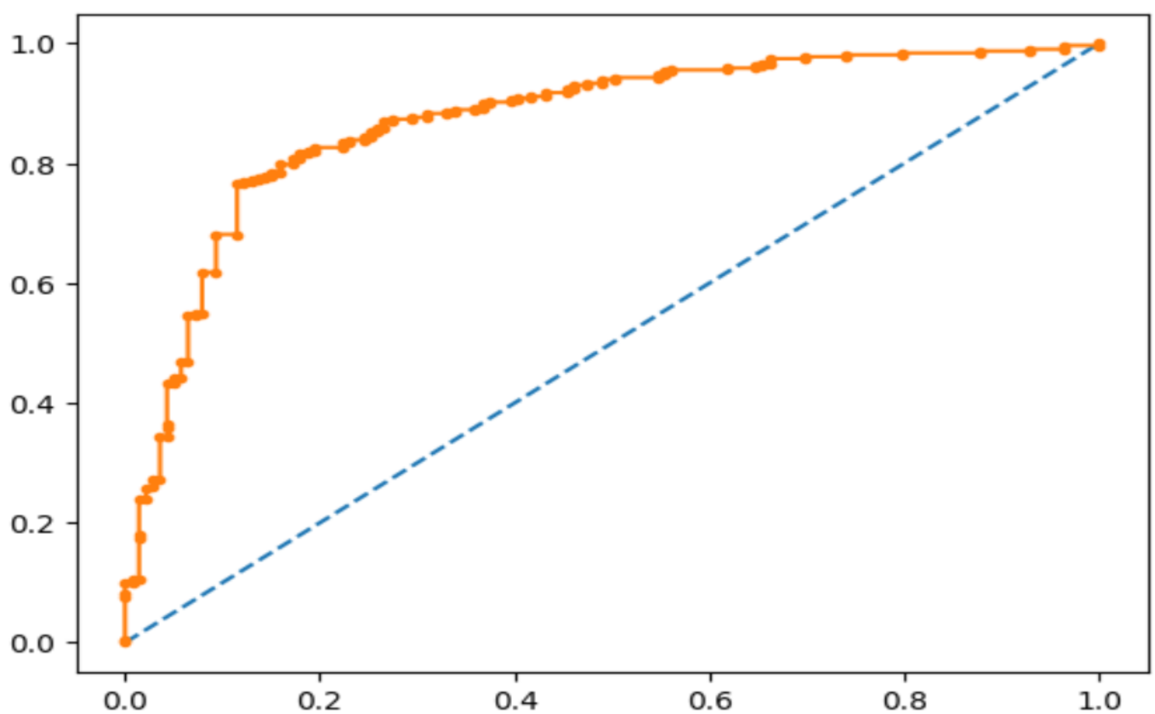
Training:

AUC: 0.998



Testing:

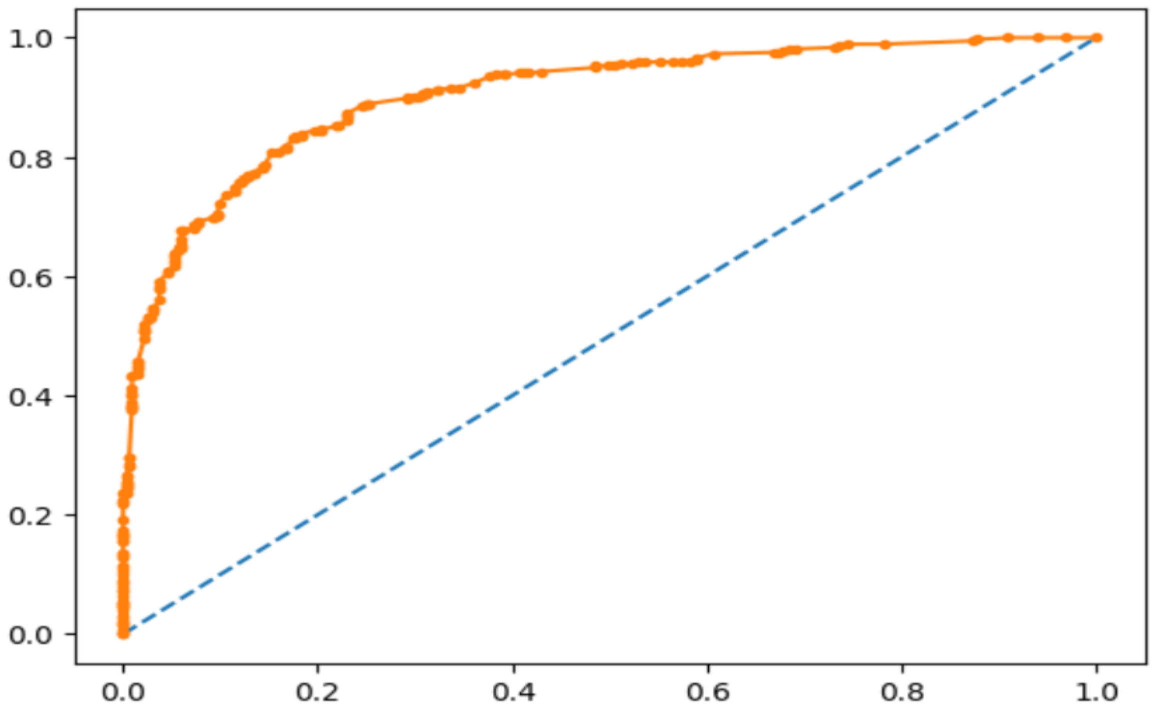
AUC: 0.871



4.6. Boosting:

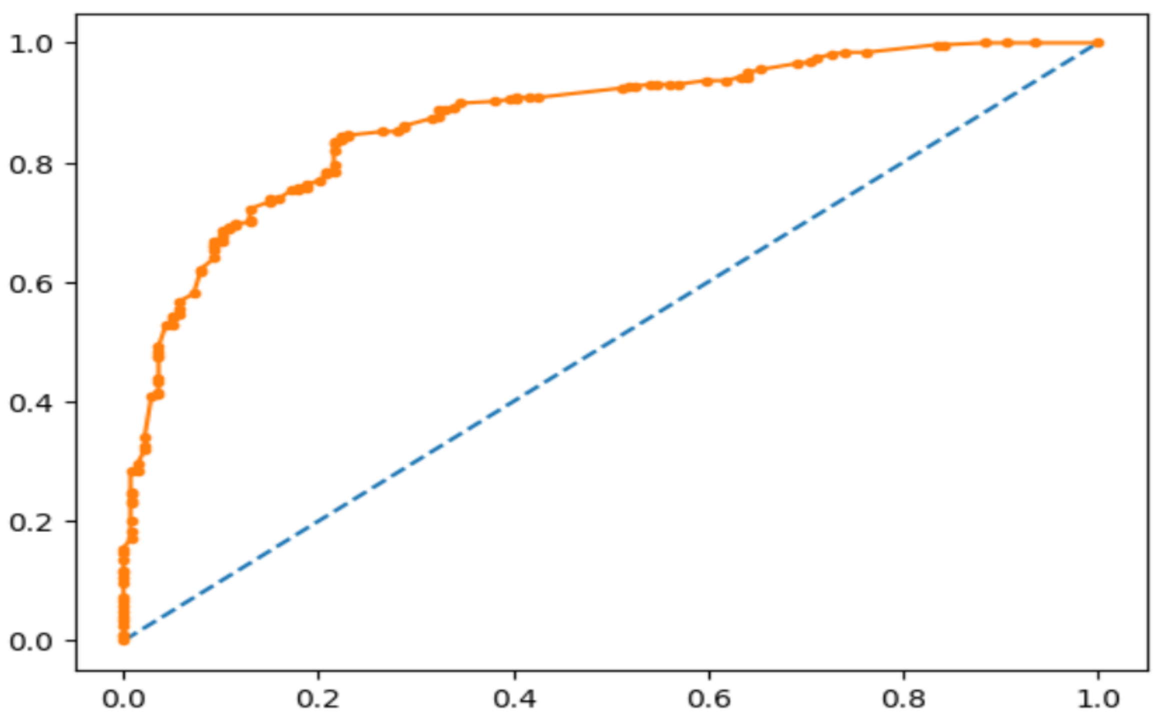
Training:

AUC: 0.904



Testing:

AUC: 0.872



Model Accuracy Comparison:

		Without Tuning	With Tuning
Logistic Regression	Training	0.85	0.85
	Testing	0.82	0.82
LDA	Training	0.84	0.84
	Testing	0.82	0.82
Naive Bayes	Training	0.84	-
	Testing	0.82	-
KNN	Training	0.87	0.88
	Testing	0.82	0.82
Bagging	Training	0.97	-
	Testing	0.82	-
Boosting	Training	0.97	-
	Testing	0.82	-

Most of the models are having overfitting problem and similar accuracies too. Overfitting is reduced after tuned. Whereas bagging model increased the training accuracy also it perform poor on testing side.
In KNN module after tuned it increased 1% of training accuracy .

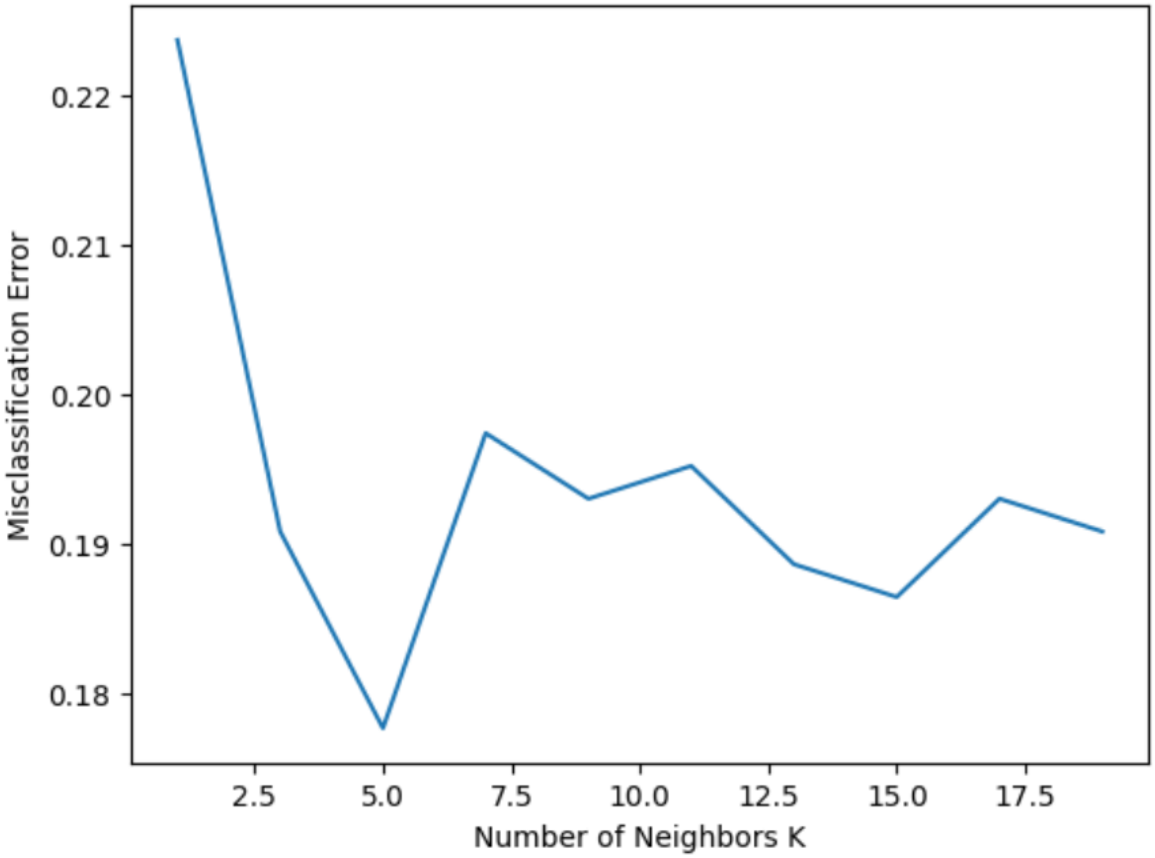
With the condition that, a model less than 10% difference in training and testing model accuracies can be considered a good model. Hence, KNN will be right model to predict voters from this given dataset.

Model f1-score comparison:
Non Tuned f1-score:

Non-Tuned		Conservative	Labour
Logistic Regression	Training	73	89
	Testing	69	87
LDA	Training	72	89
	Testing	70	87
Naive Bayes	Training	74	89
	Testing	70	87
KNN	Training	78	91
	Testing	70	87
Bagging	Training	96	98
	Testing	67	87
Boosting	Training	72	89
	Testing	69	87

Tuned f1-score:

Tuned		Conservative	Labour
Logistic Regression	Training	73	89
	Testing	69	87
LDA	Training	72	89
	Testing	70	87
Naive Bayes	Training	-	-
	Testing	-	-
KNN	Training	79	91
	Testing	70	88



This figure shows misclassification error with respect to number of neighbors k values.
It is visible that when k=5, the error is less than 0.18. It represents that the model is best than other k values.

Conclusion and Business Recommendations:
Our main business objective is- To build a model, to predict which party a voter will vote for on the basis of the given information, to create an exit poll that will help in predicting overall win and seats covered by a particular party.

- Using logistic regression model without scaling for predicting the outcome as it has the best optimized performance.
- Hyper-parameters tuning is an important aspect of model building.
- Gathering more data will also help in training the models and thus improving their predictive powers.

Problem No.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

Code Snippet to extract the three speeches:

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

Question.2.1. Find the number of Character, words in all three speeches?

Answer:

Roosevelt speech

Number of words:

	Text	totalwords
0	On each national day of inauguration since 178...	1360

Character count:

	Text	char_count
0	On each national day of inauguration since 178...	7571

Average words:

	Text	avg_word
0	On each national day of inauguration since 178...	4.539706

Number of Stopwords:

	Text	No_of_stopwords
0	On each national day of inauguration since 178...	632

Nixon Speech

Number of words:

	Text	totalwords
0	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	1819

Character counts:

	Text	char_count
0	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	9991

Average words:

	Text	avg_word
0	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	4.465091

Number of Stopwords:

	Text	No_of_stopwords
0	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	899

Kennedy Speech:

Insert the list of speech values into text column.

	Text
0	Vice President Johnson, Mr. Speaker, Mr. Chief...

Number of words:

	Text	totalwords
0	Vice President Johnson, Mr. Speaker, Mr. Chief...	1390

Character Counts:

	Text	char_count
0	Vice President Johnson, Mr. Speaker, Mr. Chief...	7618

Average words:

	Text	avg_word
0	Vice President Johnson, Mr. Speaker, Mr. Chief...	4.461871

Number of stopwords :

	Text	No_of_stopwords
0	Vice President Johnson, Mr. Speaker, Mr. Chief...	618

Question.2.2. Find most 3 common word use in all 3 speecehs?

Most frequent words coming in three speeches are:

Roosevelt speech:

```
--          22
know         9
us           8
life         6
freedom      5
speaks       5
years        5
nation       5
people       5
spirit       5
Name: count, dtype: int64
```

Roosevelt top three words used are know, us, life.

Nixon speech

```
us            25
let           22
--            17
new           15
peace         11
great          9
america        9
world.         8
america's      8
every          7
Name: count, dtype: int64
```

Nixon has top three words used are us, let, new.

Kennedy Speech

```
--          24
let         16
us          11
new          7
sides        7
pledge       7
ask          5
shall        5
cannot       4
freedom      4
Name: count, dtype: int64
```

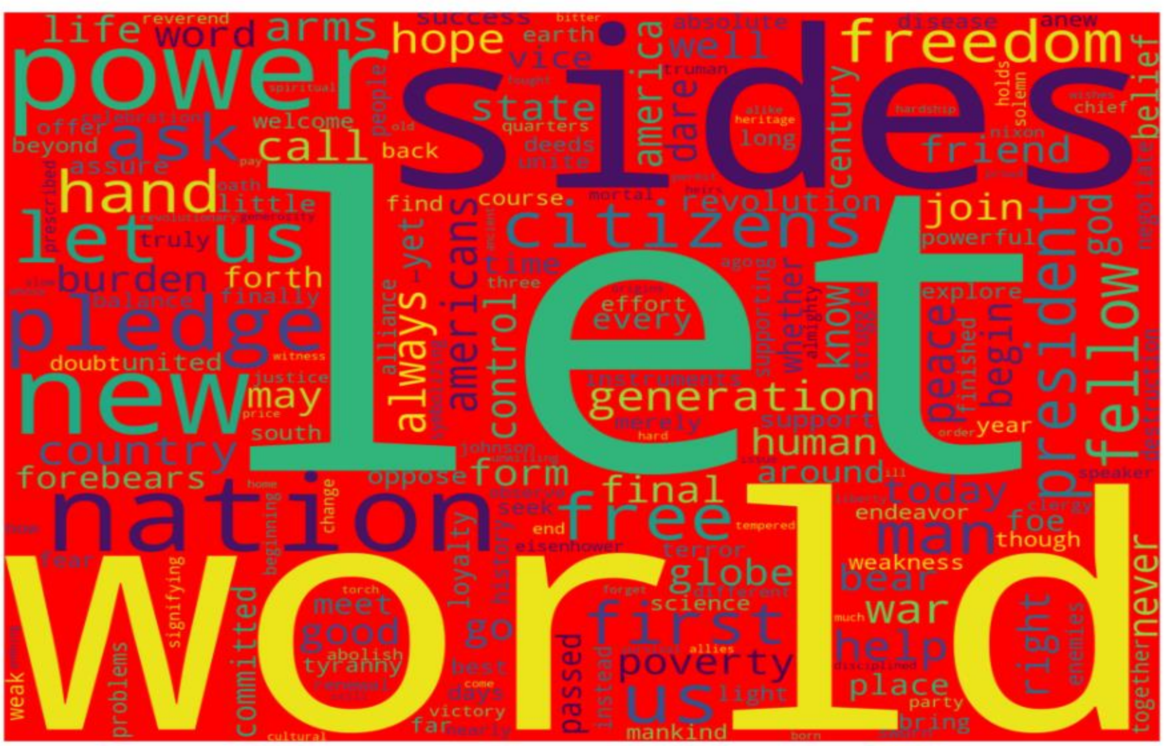
Kennedy has top three words used are let, us, new.
It shows that ‘US’ is the most common word used in all three speeches by the three presidents.

Question 2.3. Show the most common words used in all three speeches in the form of word clouds.
Answer: Using wordcloud library in python to apply the repetitive words used in speeches of president. As ‘US’, ‘World’, ‘Nation’ words are used frequently it shows the unity.

Word Cloud of Roosevelt :

Word Cloud of Nixon :

Word Cloud of Kennedy :



Conclusion:

- Our objective is to look at all 3 speeches and analyse them. To find the strength and unity of speeches.
- Hence, the output we get to see, that there are similar words present in all speeches and the word ‘nation’ is highlighted in all three speeches.