Machine Learning Graded Project

By Giridharan Velmurugan

Contents

List of Images

List of Tables

Problem 1 :-

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

1. vote: Party choice: Conservative or Labour – Target Feature
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.

[4]

1.1) 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.

About the Data Frame:

The Data Frame has Integer and Object data types. The first feature is the row number, we will set that as the index. The respective feature names are given below.

- **Object columns:** vote and gender
- **Integer columns**: age, economic.cond.national, economic.cond.household, Blair, Hague, Europe, political.knowledge.

The cell details of the data are given below. These values represent after changing the index.

- **Number of Rows/Instances:** 1525
- **Number of Columns:** 9
- **Overall size :** 13725

A sample of the data frame is shown below,

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

We do not have any null values which is a very good thing as it reduces artificial values.

We do have duplicated values in the data-frame, this is an expected situation, a lot of the features are confined to specific value ranges, hence having duplicates is inevitable while collecting the data.

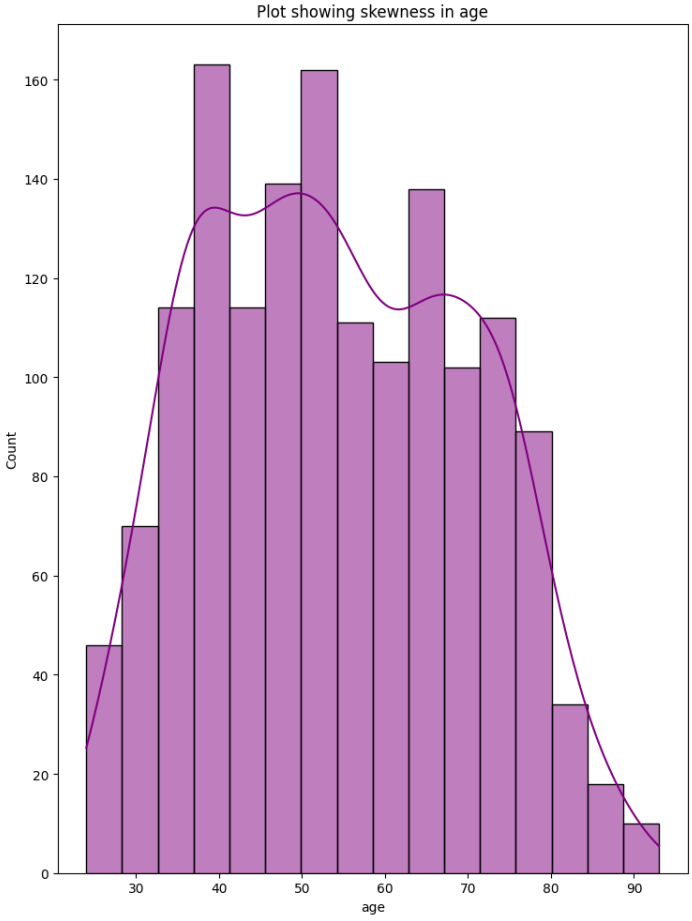
The count of duplicates is "8", which is less than 1% of the total volume. We will deal with it as required and continue with the processing of data.

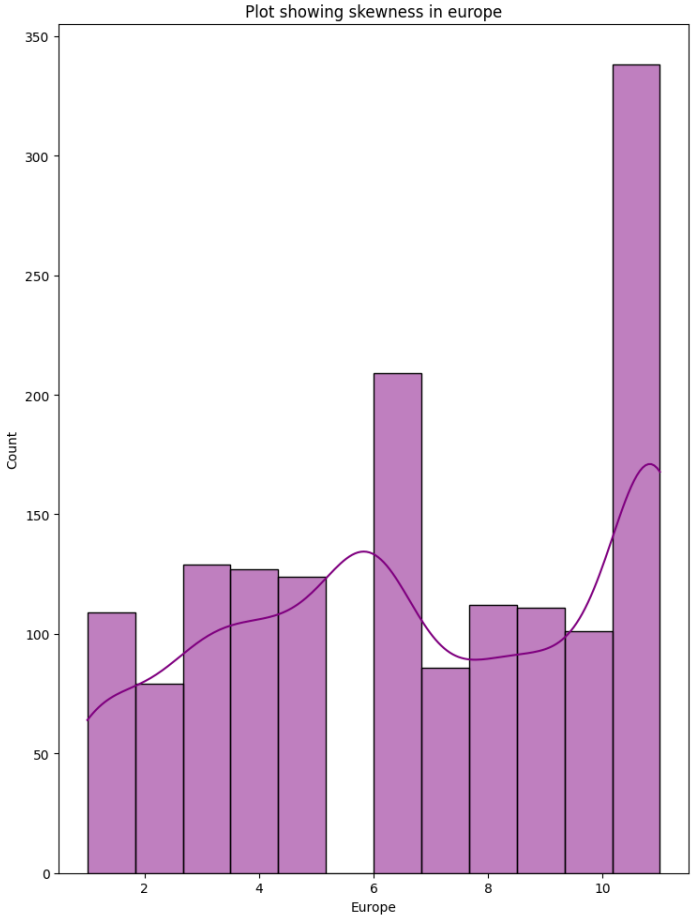
Let's see how the data frame features are skewed,

- Age: 0.145 -- Slightly skewed to the right.
- economic.cond.national: -0.240 -- Slightly skewed to the left.
- economic.cond.household: -0.150 -- Slightly skewed to the left.
- Blair -0.535 -- Heavily skewed to the left.
- Hague: 0.152 -- Slightly skewed to the right.
- Europe: -0.136 -- Slightly skewed to the left.
- political.knowledge: -0.427 -- Heavily skewed to the left.

The data features are skewed to some extent. This says that the distribution between the feature values is not normal.

The below image on the left shows the skewness of the feature 'age', we can see that the data is slightly skewed to the right.





On the right is a plot that shows the skewness of the feature 'Europe', we can see that the values are more on the right side than compared to the right. Hence the skewness on the left.

Data Statistics:

Feature name	count	mean	std	min	25%	50%	75%	max
age	1525	54.182295	15.711209	24	41	53	67	93
economic.cond.national	1525	3.245902	0.880969	1	3	3	4	5
economic.cond.household	1525	3.140328	0.929951	1	3	3	4	5
Blair	1525	3.334426	1.174824	1	2	4	4	5
Hague	1525	2.746885	1.230703	1	2	2	4	5
Europe	1525	6.728525	3.297538	1	4	6	10	11
political.knowledge	1525	1.542295	1.083315	0	0	2	2	3

The feature 'political.knowlegde' is the only one that has zero as a valid value, rest all have the minimum value as 1. A lot of the features are numerically categorical(like 5-star ratings) and are confined to a very specific integer only scale, hence the similarity in values.

Feature 'age' is the only continuous variable of the lot starting at 24 and going all the way till 93. The average age is around 55 years.

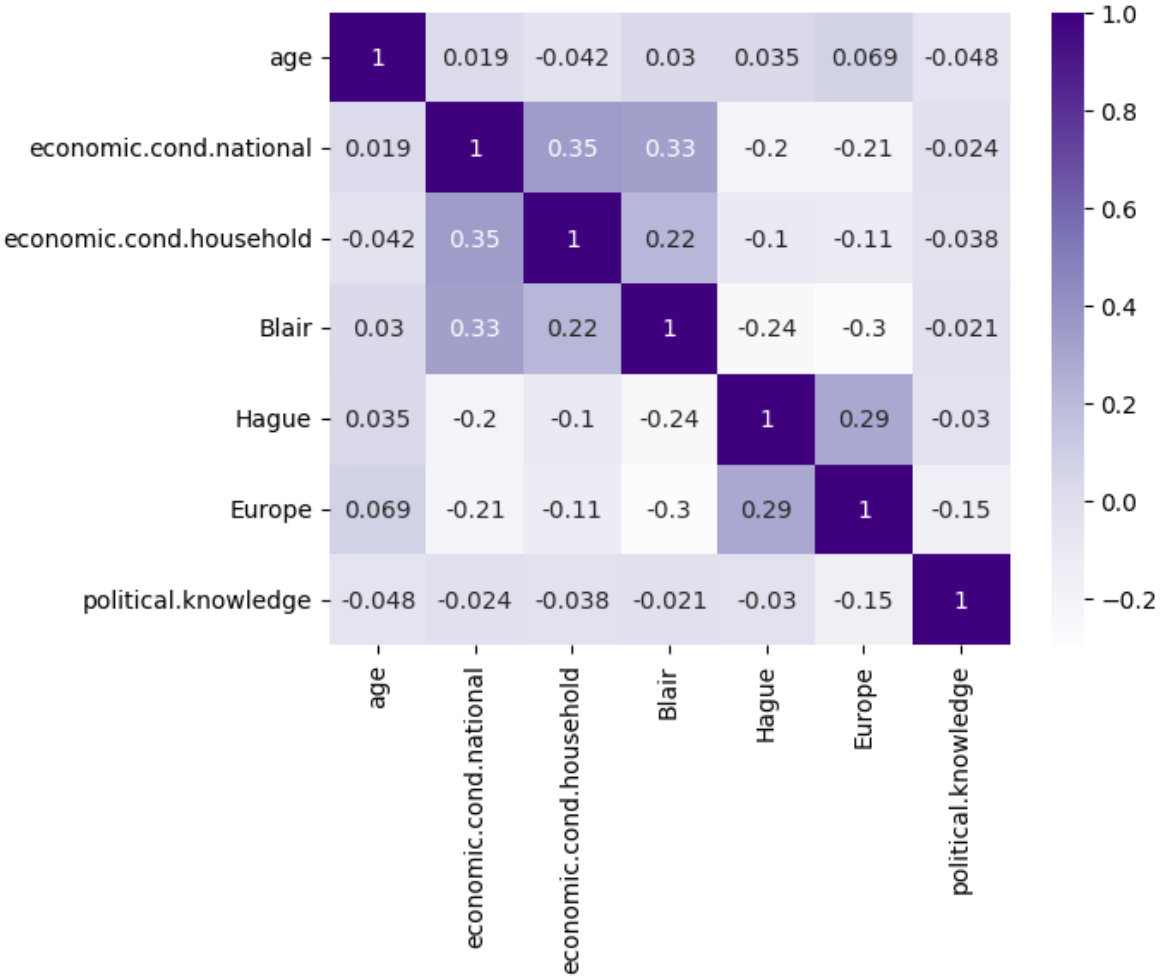
The object feature 'vote' has categories 'Labour' with a frequency of 1063 and 'conservative' with a frequency of '462'. The value counts are not balanced.

The other object feature 'gender' is more evenly spread out with categories male and female. The former has 713 counts, and the latter has 812 counts.

1.2) 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. 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.

The data frame does not have any null values, which is always a plus as it reduces the need to impute values. The data frame has 7 integer columns and 2 object columns. The object columns are 'vote' (which is our target variable) and 'gender'. The data frame has 1525 record instances.

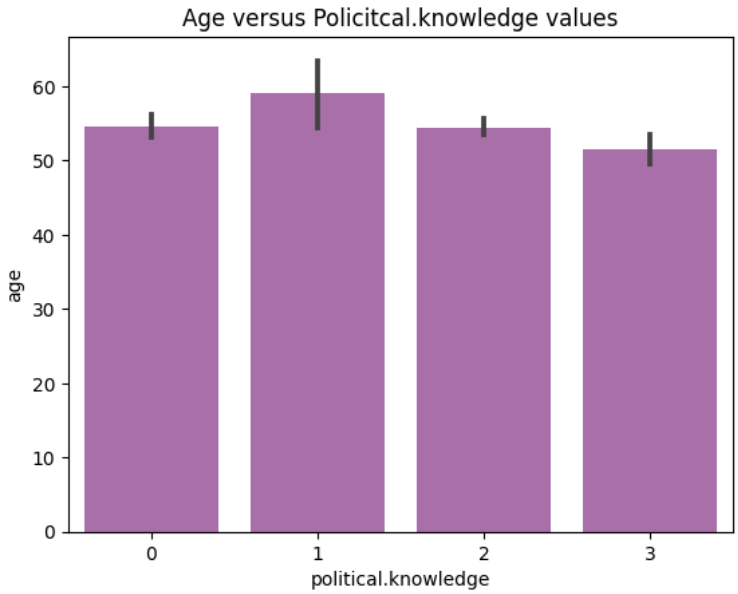
[8]



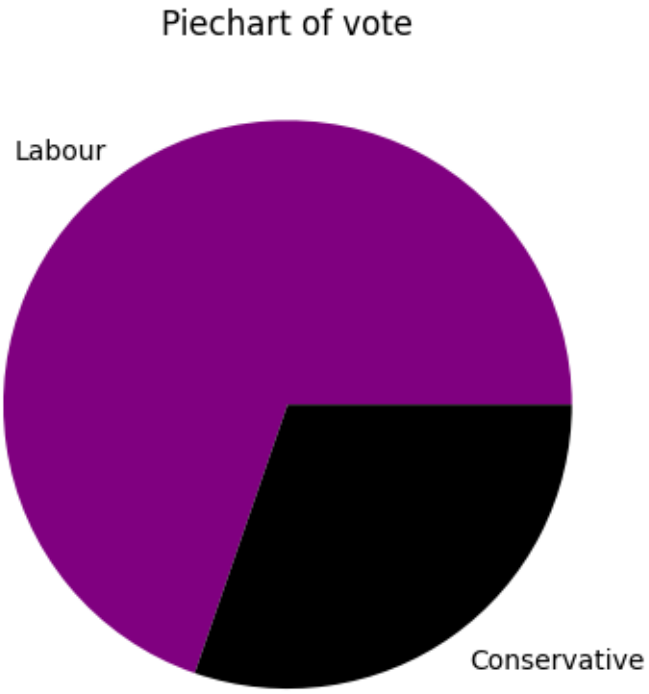
The above heatmap shows the correlation between the features, it is safe to assume that the values are not correlated to a significant extent. None of the feature combinations cross a value of 0.4 mark, the maximum is 0.35, barring the diagonal values that is the feature against itself.

Feature economic.cond.national when paired against economic.cond.household shows our maximum correlation at 0.35. All the other pairing come under this mark.

The data frame has features that can be considered as independent to each other.



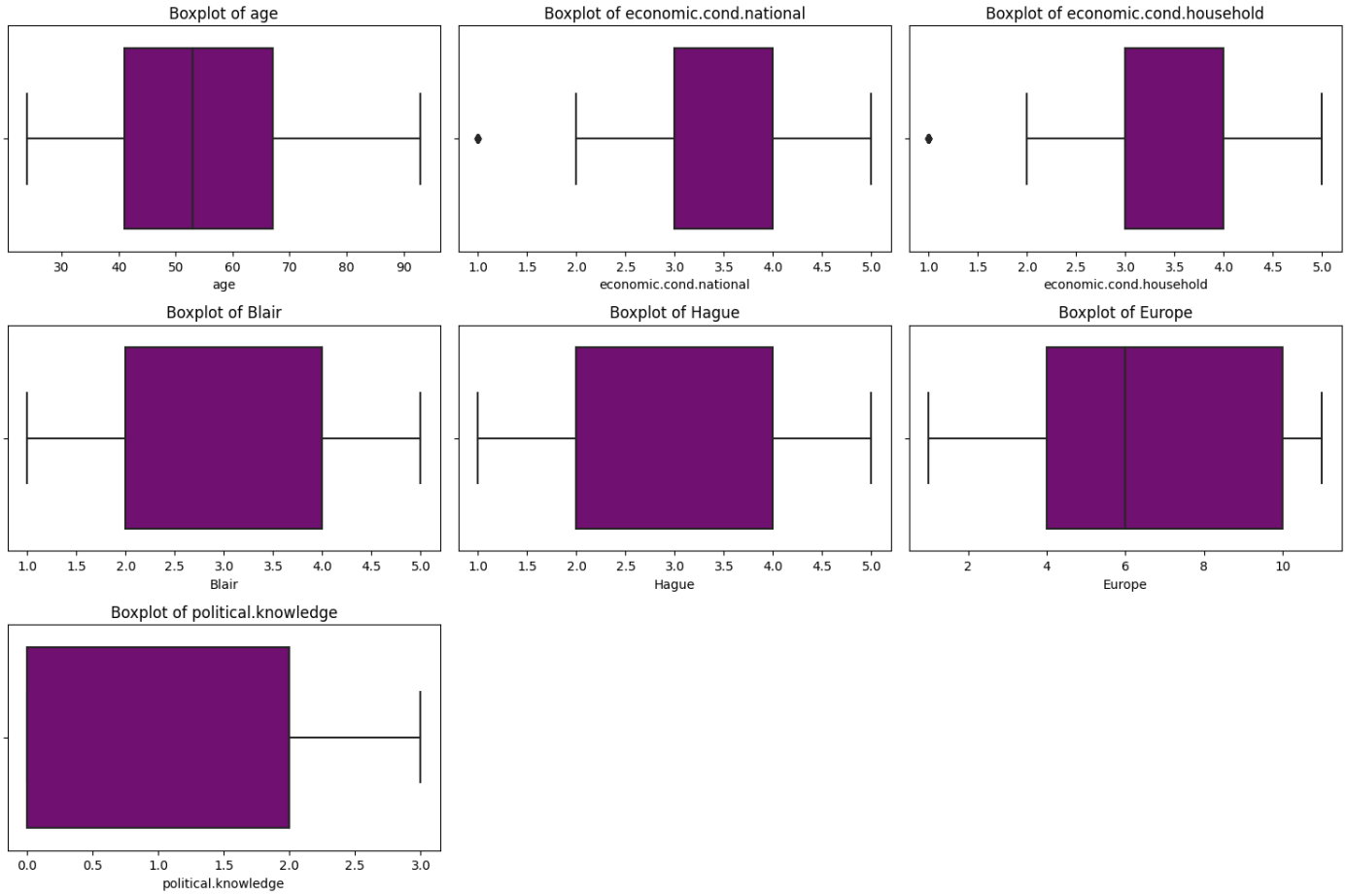
The above plot shows the age distribution with respect to the political knowledge levels. The black wicks on the top show the ranges with which the age varies, the value '1' ranges anywhere between 55 to over 60 years. The distribution is evenly spread and we do not have any one value overpowering others.



The pie-chart on the left shows the distribution of the values of the feature 'vote'. Here we can see that the value Labour is around 2/3rd of the total volume.

This in a way will hinder when prepare models, we have a comparatively less data to label a particular instance as 'conservative'.

Boxplots to see if we have any outliers:- The boxplots below are only for numerical features.

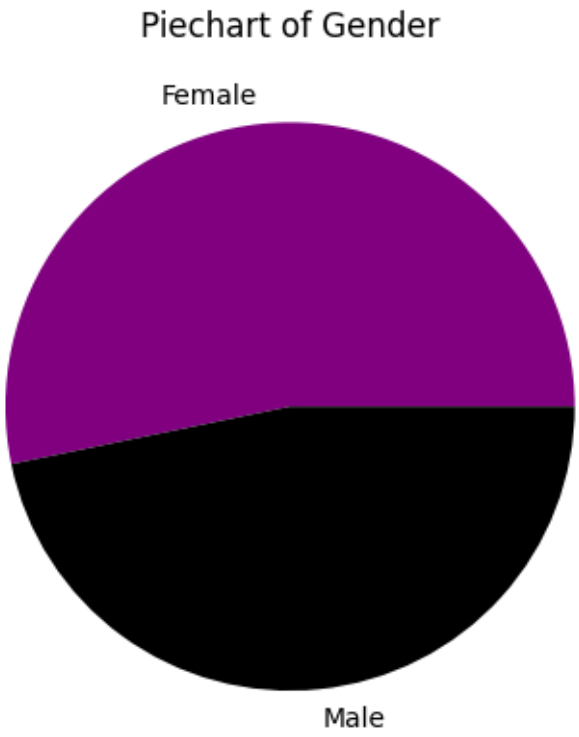


The above boxplots show that we have outliers only in the features 'economic.cond.national' and 'economic.cond.household'. All those outliers all point to the value '1', what we have to understand is that those values are all acceptable values. They are shown as outliers because they are very small in frequency as compared to other values and are also comparatively far from the mean.

The surprising thing is that, the feature 'age' does not have any outliers, the maximum age of 93 years and the minimum age of 24 years doe not become parts of outliers.

The features 'Blair' and 'Hague' have very similar distribution, the same is the case for features 'economin.cond.national' and 'economic.cond.household'. This particular characteristic is because the values are not continuous and are restricted to a value range.

'Europe' and 'political.knowledge' do not have any outleirs, and the latter is more spreadout of the latter is not even and focussed on the initial values, which is the opposite when compared to the former feature.



The above piechart shows the volume composition of the feature 'gender'. The values are very close, the female count just edges over the male count by a very thin margin.

1.3) 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.

We will remove the duplicates from this step and the total record instances will recude to 1517.

Encoding the data – We will encode the object features 'vote' and 'gender'.

- Labour will be set to 0, and conservative to be 1
- Male to be 0 and female to be 1

A view of the data frame after the change is given below.

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

Scaling

Scaling is done to obtain a more efficient and better performing model, some algorithms are more robust while some are very sensitive. Scaling is paramount when it comes to distance based algorithms. For the dataset that we have here, scaling is required as we have age in a larger scale compared to other features.

One might contest that the scales are not huge and the differences does not warrant the need for scaling. Distance based models when they aggregate different features values, even the thinnest of margins will impact the performance characteristics of the model. Hence scaling is pre-requisite when we opt for the said concepts.

A data preview is given below,

	vote	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	gender
0	1.0	0.275362	0.50	0.50	0.75	0.00	0.1	0.666667	0.0
1	1.0	0.173913	0.75	0.75	0.75	0.75	0.4	0.666667	1.0
2	1.0	0.159420	0.75	0.75	1.00	0.25	0.2	0.666667	1.0
3	1.0	0.000000	0.75	0.25	0.25	0.00	0.3	0.000000	0.0
4	1.0	0.246377	0.25	0.25	0.00	0.00	0.5	0.666667	1.0
5	1.0	0.333333	0.50	0.75	0.75	0.75	0.3	0.666667	1.0
6	1.0	0.478261	0.25	0.25	0.75	0.75	1.0	0.666667	1.0
7	1.0	0.768116	0.50	0.75	0.75	0.00	0.0	0.000000	1.0
8	1.0	0.217391	0.50	0.50	0.75	0.75	1.0	0.000000	0.0
9	1.0	0.666667	0.50	0.25	1.00	0.00	1.0	0.666667	1.0

Lets see the summary statistics of the dataframe after scaling

Feature Name	mean	std	min	25%	50%	75%	max
age	0.438279	0.227561	0.0	0.246377	0.420290	0.623188	1.0
economic.cond.national	0.561305	0.220448	0.0	0.500000	0.500000	0.750000	1.0
economic.cond.household	0.534443	0.232767	0.0	0.500000	0.500000	0.750000	1.0
Blair	0.583883	0.293693	0.0	0.250000	0.750000	0.750000	1.0
Hague	0.437376	0.308120	0.0	0.250000	0.250000	0.750000	1.0

Feature Name	mean	std	min	25%	50%	75%	max
Europe	0.574028	0.329904	0.0	0.300000	0.500000	0.900000	1.0
political.knowledge	0.513514	0.361472	0.0	0.000000	0.666667	0.666667	1.0

We can see a difference between the values of scaled and non-scaled data. Mainly the mean and standard decviation of the feature 'age' is much more in scale when compared to mean and standard deviation of the other features, which was not the case before.

Here we have used -min-max scaling technique to preserve the distribution of each of the features.

Dataframe is split into two parts, one for training and the other for testing, in the ratio of 70(training) : 30(test). We will also make sure that the distribution of train and test dataset is not random, but based upon the ration of vote categories, we will make sure the distribution is mimicked throughout.

Please find the value counts below.

Training dataset :

- Conservative - 739
- Labour - 322

Test dataset :

- Conservative - 318
- Labour - 138

1.4) 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 should be shared if any custom changes are made to the parameters while building the model. Calculate Train and Test Accuracies for each model. Comment on the validness of models (over fitting or under fitting).

Logistic regression

Applying Logistic regression to the data's that we have prepared, we get the accuracy score for the train data to be 83.69%, which is a good score. The same model for the test data gave an accuracy score of 83.77%. This suggests that the model is not overfitting the training data and is able to generalize well to unseen data.

The model was set with a max limit of 10000 iterations to allow it to converge. This model was also set without any penalty(none was used as penalty value).

	Class	Precision	Recall
Train	0(Labour)	0.77	0.66
	1(Conservative)	0.87	0.91
Test	0(Labour)	0.77	0.66
	1(Conservative)	0.86	0.92

Precision and recall metrics for both test and train datasets from the table above show that we have a fairly good model but still needs improvement. The precision is at 77% for both test and train for the '0' class and for the '1' class there is a difference of 1unit between train and test. This means that 77% of the predictions for class 0 are actually positive, appling same 87% of the predictions of class 1 are actually positive. This is not the case for recall, the precision stays constant, but the recall improves by 1unit between the datasets used. From recall we can also deduce that class '1' will be identified correctly more that compared to class '0' buy a good margin. This means that 77% of the predictions for class 0 are actually positive.

Linear Discriminant Analysis

Applying Linear Discriminant Analysis to the data's that we have prepared, we get the accuracy score for the train data to be 83.78%, which is a good score. The same model for the test data gave an accuracy score of 84.21%. This suggests that the model is not overfitting the training data and is able to generalize well to unseen data. LDA was not tampered with any configuration, it was run as default.

	Class	Precision	Recall
Train	0(Labour)	0.76	0.68
	1(Conservative)	0.87	0.91
Test	0(Labour)	0.78	0.67
	1(Conservative)	0.87	0.92

Similar to logistic regression precision and recall metrics for both test and train datasets from the table above show that we have a fairly good model but still needs improvements in both aspects. The precision is at 76% for both train and increses by 2 units for test for the '0' class and for the '1' class both test and train datasets produced the same score. This means that between 76% and 78% the predictions for class 0 are actually positive, likewise for class '1' it stays constant at 87%. Like logistic regression there is a good difference In terms of recall, the model is able to identify class '1' more efficiently that class '0'.

For both of the models, recall is one sided to class ' 1' representing 'Conservative' vote. This can be attributed to the fact that the data avialable is providing more information about class '1' than compared to class '0'.

1.5) Apply KNN Model and Naïve Bayes Model (2pts). Interpret the inferences of each model (2 pts). Successful implementation of each model. Logical reason should be shared if any custom changes are made to the parameters while building the model. Calculate Train and Test Accuracies for each model. Comment on the validness of models (over fitting or under fitting)

Gaussian Naïve Bayes

Applying Gaussian Naïve Bayes algorithm to the data's that we have prepared, we get the accuracy score for the train data to be 82.46 %, which is a good score. The same model for the test data gave an accuracy score of 84.42%. This suggests that the model is not overfitting the training data and is able to generalize well to unseen data. Gaussian NB is a simpler model and is not configurated is anyway.

	Class	Precision	Recall
Train	0(Labour)	0.72	0.70
	1(Conservative)	0.87	0.88
Test	0(Labour)	0.76	0.71
	1(Conservative)	0.88	0.90

Precision and recall metrics for both test and train datasets from the table above show that we have a fairly good model but still needs improvement, seen similarly to logistic regression and Linear Discriminant Analysis. The precision is at 72% for train data, but jumps by 4 units to 76% for the test data with respect to class '0' and for the '1' class there is a difference of 1 unit between train and test.

Recall for class 1 between test and train has a difference of 2 unit, with test dataset taking the upper hand, class '0' only improves by 1unit. The difference between the classes itself is comparatively reduced, in the sense that Gaussian Naïve bayes is able to identifiy class '0' better than compared to Logsitic regression or LDA.

K-Nearest Neighbors

Applying K-Nearest Neighbors algorithm to the data's that we have prepared, we get the accuracy score for the train data to be 82.84%, which is a good score. The same model for the test data gave an accuracy score of 84.42%. This suggests that the model is not overfitting the training data and is able to generalize well to unseen data.

The K value for the model was set at '3'. Which is a very common place to start any KNearest Neighbors setup. The default distance metric was used(default will generally point to minkowski distance metric).

	Class	Precision	Recall
Train	0(Labour)	0.72	0.70
	1(Conservative)	0.87	0.88
Test	0(Labour)	0.76	0.71
	1(Conservative)	0.88	0.90

The precision and recall scores obtained are identical to what we obtained in Gaussian Naïve Bayes model. This is a surprising coincidence.

The K-Nearest Neighbors model is also able to perform better at identifying class '0' like Guassian NB.

The parity in recall values for both the classes in terms of test and train datasets can be attributed to the difference in vote categories counts. An significant abundance of class '1' is resulting in models that can predict it better than class '0'.

1.6) 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. 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.

Modeul tuning is always a norm in machine learning, Each data frame has different characteristics and accordingly each model has to be tweaked to better suit the data.

Logistic Regression

We have used "GRID SEARCH" to find the best parameters that might help make logistic regression a better performer. The following parameters were selected and tried.

- **'C': [0.001, 0.01, 0.1, 1, 10, 100]** → C is a parameter used to control the regularization strength
- **'penalty': ['l2', None]** → this is the regularization parameter that will put on the weights when required.
- **'solver': ['newton-cg', 'lbfgs', 'sag', 'saga']** → the solution provider
- **'max_iter': [10000]** → the maximum iterations count that a model can take to converge.
- **'tol': [0.001, 0.0001, 0.00001]** → This is the stopping criteria.

Best paramters from grid search : {'C': 1, 'max_iter': 10000, 'penalty': 'l2', 'solver': 'newton-cg', 'tol': 0.001}

Now when we apply those parameter values to our model and run them we get the following resutls.

- Train accuracy – 83.60%
- Test accuracy – 83.55%

Model features Coefficients,

- Age → (-1.21)
- economic.cond.national → (1.39)
- economic.cond.household → (0.47)
- Blair → (2.21)
- Hague → (-2.18)
- Europe → (-2.19)
- Political.knowlegde →(-1.04)
- Gender → (0.003)

The model coefficients show that Blair, Europe and Hague are the top three most influential features.

The precision and recall are as follows, they are nearly identical to the baseline model of logistic regression that we have discusssed previoulsy in section 1.4.

	Class	Precision	Recall
Train	0(Labour)	0.77	0.65
	1(Conservative)	0.86	0.92
Test	0(Labour)	0.77	0.64
	1(Conservative)	0.86	0.92

Linear Discriminant Analysis

LDA being a different model will have different model parameter(s) that can be tweaked. The same is given below

- **'solver': ('svd', 'lsqr', 'eigen')** → Solver is the core algorithm that will be used to arrive at the classifications requried.

The best parameter for LDA was {'solver': 'svd'}

Now when we apply those parameter values to our model and run them we get the following resutls.

- Train accuracy – 83.78%
- Test accuracy – 84.21%

Model features Coefficients

- Age → (-1.73)
- economic.cond.national → (1.58)
- economic.cond.household → (0.52)
- Blair → (-2.94)
- Hague → (-3.49)
- Europe → (-2.56)
- Political.knowlegde →(-1.48)
- Gender → (-0.03)

The model coefficients show that Hague, Blair and Europe are the top three most influential features in order.

The precision and recall are as follows, they are nearly identical to the LDA mdel that we have discusssed previoulsy in section 1.4. The recall for class '0' in the train dataset alone has increased by 2 units.

	Class	Precision	Recall
Train	0(Labour)	0.76	0.68
	1(Conservative)	0.87	0.91
Test	0(Labour)	0.78	0.67
	1(Conservative)	0.87	0.92

Gaussina Naïve Bayes

The method grid search was also applied for Gaussian NB to find the better suited parameters. The following were tested.

- **'var_smoothing': [1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3]** → In GNB, we assume that the features are normally distributed. Sometimes, however, a feature might have a variance that is zero or close to zero, which can cause issues in calculations. The numbers are the values that will be added to the variances of the features. A smaller value like 1e-9 will add very little smoothing, keeping the model closer to the true distribution of the features. A larger value like 1e-3 will smooth out the feature's distribution more, making the model more robust but potentially less accurate for this specific dataset.

Best paramters: {'var_smoothing': 1e-09} → this is a good, as the face value is very minimal.

Applying the var_smoothing value to GNB we get the following scores and resutls.

- Train accuracy – 82.46%
- Test accuracy – 84.42%

The precision and recall are as follows, they are identical to the Gaussian Naïve Bayes model that we have discusssed previoulsy in section 1.5. The var_smoothing value, being very small does in a way mean that the data features distribution is already well balanced and performing optimally.

	Class	Precision	Recall
Train	0(Labour)	0.72	0.70
	1(Conservative)	0.87	0.88
Test	0(Labour)	0.76	0.71
	1(Conservative)	0.88	0.90

K-Nearest Neighbors

Unlike Logistic regression of LDA, K-Nearest Neighbors model is one where we do not have a equation to guide the final classification. As such this makes it a more hand on model.

Grid search with the following parameters was applied to find a optimal predictor.

- **'n_neighbors': [3, 5, 7, 9]** → This is the K value, which says how many neighbors we should consider
- **'metric': ['minkowski', 'manhattan', 'euclidean']** → this is the distance metric that guides the classification part.

Best Parameters based on grid search for KNN are {'metric': 'minkowski', 'n_neighbors': 9}

Applying the grid search parameters to KNN, we get the following results.

- Train accuracy – 83.60%
- Test accuracy – 83.55%

The precision and recall are as follows, Class '0' has its train dataset precision increased by 5 units which is a very good. Precision has improved in all fronts.

Recall has also improved for the train class '0', but we can see that the recall for class '1' has taken a hit in both the data cases. This means that the metric is able to work out a comparatively better identifying class '0' model.

	Class	Precision	Recall
Train	0(Labour)	0.77	0.65
	1(Conservative)	0.86	0.92
Test	0(Labour)	0.77	0.64
	1(Conservative)	0.86	0.92

As of now, we have only attemped direct techniques, now we will work our way with ensemble techniques.

Random Forest Classifier

Random forest classifier is a collection of decision trees that work together to classificy data. By this we can get a more robust and more accurate model in all fronts.

Using Grid search on the following parameters to get the optimal random forest classifier model

- **'n_estimators': [50, 100, 200]** → The n_estimators parameter in a Random Forest algorithm refers to the number of trees in the forest. Essentially, it controls the size of the ensemble. Each tree in the forest is a decision tree that is trained on a subset of the data and/or features, and the Random Forest algorithm combines the outputs of all these trees to make a final prediction.
- **'max_depth': [None, 10, 20, 30]** → this is the parameter that controls the depth of the tree branching down.

Best Parameters for Random Forest: {'max_depth': 10, 'n_estimators': 200}

Using the grid search parameters we develop a new Random Forest Classifier model, we get the following results.

- Train accuracy – 96.70%
- Test accuracy – 83.55%

There is a very huge difference in the accuracy scores, the train is working very well, but the model is being weak to the test data.

	Class	Precision	Recall
Train	0(Labour)	0.95	0.94
	1(Conservative)	0.97	0.98
Test	0(Labour)	0.75	0.68
	1(Conservative)	0.87	0.90

The recall and precision values show us a very clear picture that the model is overfitting the data. The model buckles when we ask it classificy unseen data. The closest value is the recall for class '1' for the test dataset is down by 8 units when compared to the recall of class '1' in the train dataset. Although the test perofrmance is on par with the othe models that we have here it is still not worthy of compute effort that may require.

Bagging classifier

Bagging classifier is another ensemble technique that aims to reduce the variance of a model by averaging multiple predictions.It takes multiple subsets of the original dataset by sampling with replacement and trains a base model on each subset. The final prediction is an average (regression) or majority vote (classification) of the predictions from each base model. Bagging classifier will use Random forest as the estimator.

Each model is independent of the others, so they can be trained in parallel, making bagging computationally efficient. Bagging classifier uses a decision tree model when it is run as default, but we can ask to run other models. For now we will only be using a standard bagging classifier.

We will apply grid search on bagging classifier and the following parameters were played upon.

- **'n_estimators': [10, 20, 30,]** → n_estimators is similar to what we saw in random forest classifier
- **'max_samples': [0.5, 1.0]** → This parameter controls the size of the subsets that are created for training each base estimator. It can be expressed as an integer or a float.
- **'bootstrap': [True, False]** → This is a boolean parameter that indicates whether samples are drawn with replacement (True) or without replacement (False).

Best Parameters: {'bootstrap': False, 'max_samples': 0.5, 'n_estimators': 20}

Applying the grid search parameters and creating a new bagging classifier we get the following scores.

- Train accuracy – 94.53%
- Test accuracy – 84.64%

There is a very huge difference in the accuracy scores, the train is working very well, but the model is being weak to the test data.

	Class	Precision	Recall
Train	0(Labour)	0.93	0.86
	1(Conservative)	0.94	0.97
Test	0(Labour)	0.75	0.66
	1(Conservative)	0.86	0.91

The recall and precision values show us a very clear picture that the model is overfitting the data. Similar to the Random forest classifier model that we just discussed above. The model buckles when we ask it classificy unseen data.

The difference in train and test metrics is to some extent less than compared to the previous model. Precision and recall is better in train than test data. A classic case of overfitting.

Gradient Boosting

Boosting aims to improve the model by training on the errors made by the ensemble. It starts with a weak model and iteratively adds more models, adjusting the weights of instances that were misclassified in previous rounds. This way, the new model focuses more on the "hard-to-classify" instances.

AdaBoost, Gradient Boosting, and XGBoost are popular boosting algorithms. Because each model is dependent on the errors of its predecessor, boosting models are trained sequentially, which can be computationally expensive.

For boosting too we will follow the same process, below are the grid search parameters chosen to play,

- **'n_estimators': [50, 100, 150]** → n_estimators is similar to what we saw in random forest classifier.
- **'learning_rate': [0.01, 0.1, 0.5]** → This parameter scales the contribution of each tree to the final prediction.
- **'max_depth': [3, 4, 5]** → The maximum depth of the individual decision trees.

Best Parameters for Gradient Boosting: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 50}

Applying the grid search parameters and creating a new boosting classifier we get the following scores.

- Train accuracy – 88.12%
- Test accuracy – 83.77%

There is a very huge difference in the accuracy scores, the train is working very well, but the model is being weak to the test data.

	Class	Precision	Recall
Train	0(Labour)	0.85	0.75
	1(Conservative)	0.89	0.94
Test	0(Labour)	0.77	0.67
	1(Conservative)	0.86	0.91

The recall and precision shows us that the model is performing way more consistently than the previous ensemble techniques.

We can see a drop in the overll precison and recall values, but the difference in terms of recall for class '1' is still very closer now.

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, 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)

Logistic regression

Below table provides the comprehensive look

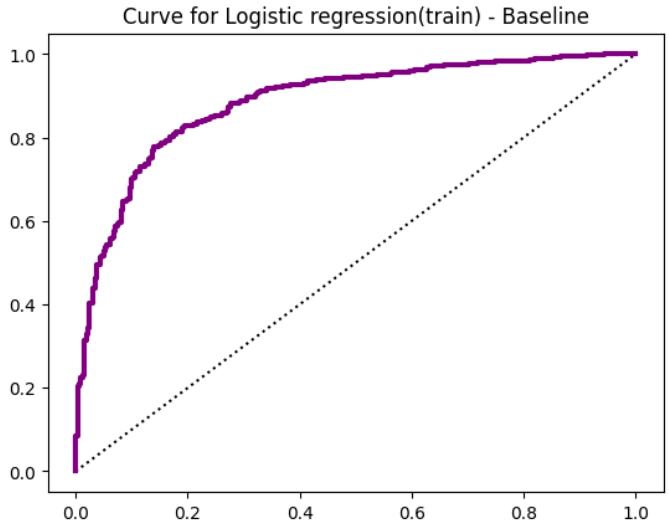
Model type	T/T	Accuracy	Precision		Recall		AUC score	True (-)	False (+)	False (-)	True (+)	F1 score	
			Class 0	Class 1	Class 0	Class 1						Class 0	Class 1
Baseline Model	Train	83.69	0.77	0.87	0.66	0.91	0.885	213	109	64	675	0.71	0.89
	Test	83.77	0.77	0.86	0.66	0.92	0.897	91	47	27	271	0.71	0.89
Grid_search Model	Train	83.60	0.77	0.86	0.65	0.92	0.885	210	112	62	677	0.71	0.89
	Test	83.55	0.77	0.86	0.64	0.92	0.896	89	49	26	292	0.71	0.89

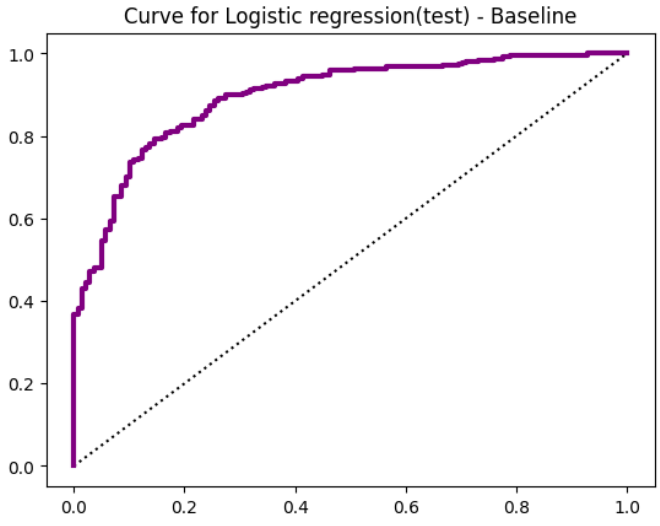
Both the Baseline and Grid_search Models have similar accuracy (~83%) for both training and test sets. The F1 score is the harmonic mean of precision and recall. For Class 0, it's around 0.7, and for Class 1, it's around 0.89. This shows that the model is better at identifying Class 1 than Class 0.

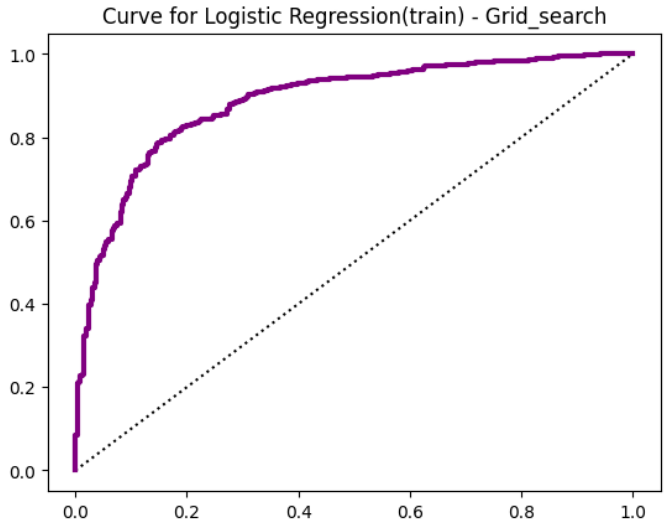
Both models have an AUC score close to 0.89, which is very good. This metric tells us that the models do well in distinguishing between the classes. We can see a slight jumble of the confusion matrix values but they finally retain similar scores.

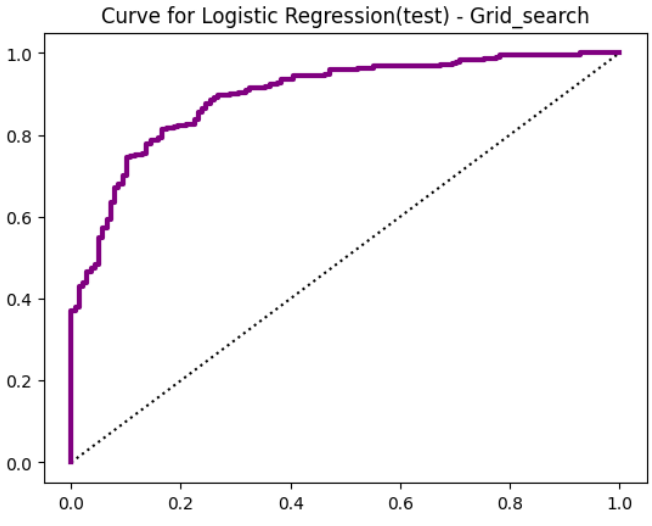
Both models have similar performance metrics on the training and test sets, indicating that they're neither overfitting nor underfitting. Both models are better at identifying Class 1 instances over Class 0, as indicated by higher recall and F1 scores for Class 1. The Grid_search Model doesn't show significant improvement over the Baseline Model.

Please find the Curves, below for both the models









The plots are similar, the train dataset is able to have a smoother curve, while the test dataset is much more rough at the middle, but smoothens out at the end sections. The curves also show that the model is not overfitting.

X-axis is the False Positive Rate, Y-axis is the True Positive Rate. The Diagonal Line is the randome classifier. The Curve is our classifier, a perfect classifier would go straight up to the top-left corner, giving an AUC of 1.0. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.

A rough or jagged curve may indicate that the model's performance varies significantly with different thresholds.A rough curve could be a sign that the model is overfitting to the noise in the data, capturing random fluctuations as if they were meaningful patterns.

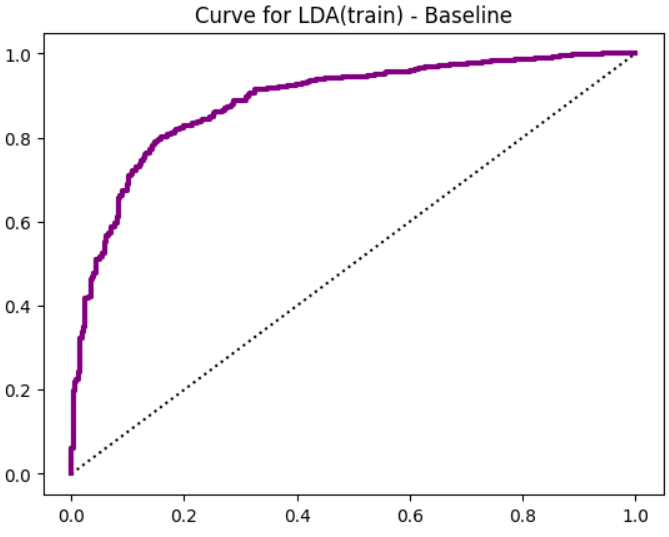
Linear Discriminant Analysis

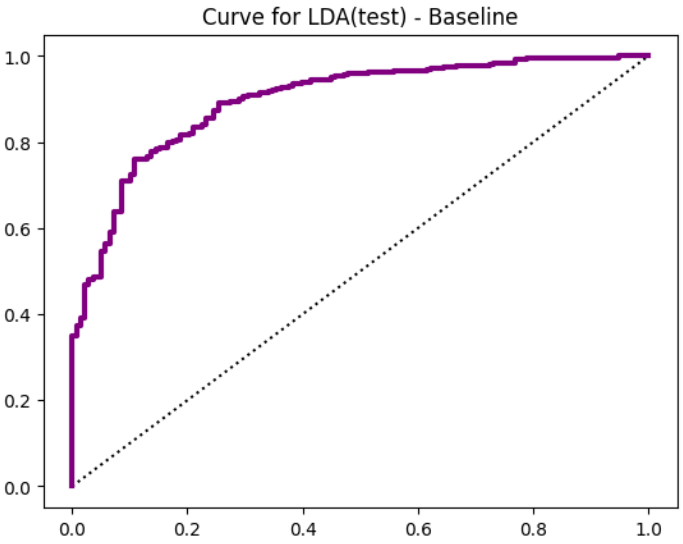
The below table is a comprehensive look of the models.

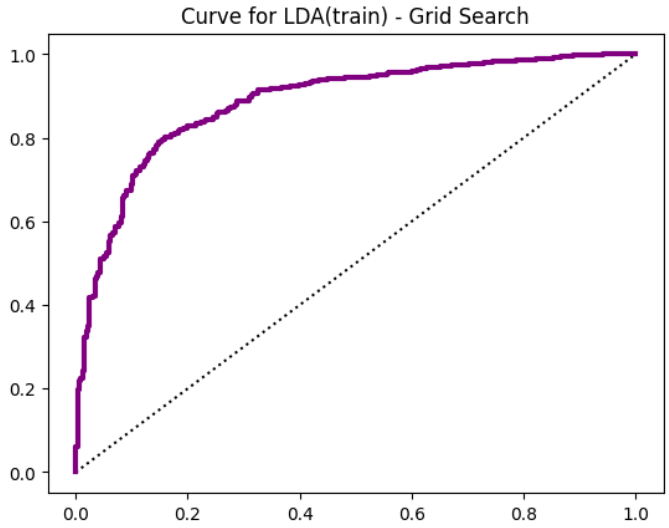
Model type	T/T	Accuracy	Precision		Recall		AUC score	True (-)	False (+)	False (-)	True (+)	F1 score	
			Class 0	Class 1	Class 0	Class 1						Class 0	Class 1
Baseline Model	Train	83.78	0.76	0.87	0.68	0.91	0.885	219	103	69	670	0.72	0.89
	Test	84.21	0.78	0.87	0.67	0.92	0.896	93	45	27	291	0.72	0.89
Grid_search Model	Train	83.77	0.76	0.87	0.68	0.91	0.885	219	103	69	670	0.72	0.89
	Test	84.21	0.78	0.87	0.67	0.92	0.896	93	45	27	291	0.72	0.89

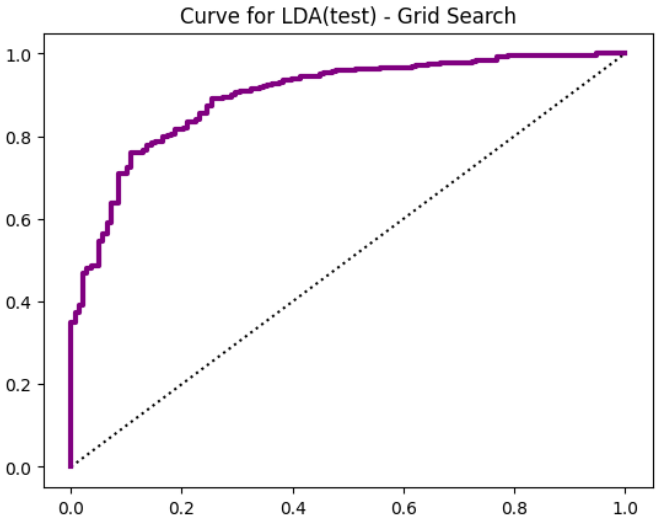
The shows that the baseline model and the grid_search model are identical in all aspects.

Both models have similar performance metrics on the training and test sets, indicating that they're neither overfitting nor underfitting. Both models are better at identifying Class 1 instances over Class 0 as seen in logistic regression and as indicated by higher recall and F1 scores for Class 1. The Grid_search Model doesn't show any improvement over the Baseline Model.









The plots are similar, the train dataset is able to have a smoother curve, while the test dataset is much more rough at the middle, but smoothens out at the end sections. The curves also show that the model is not overfitting.

The plots are also similar to Logistic regression curves as well.

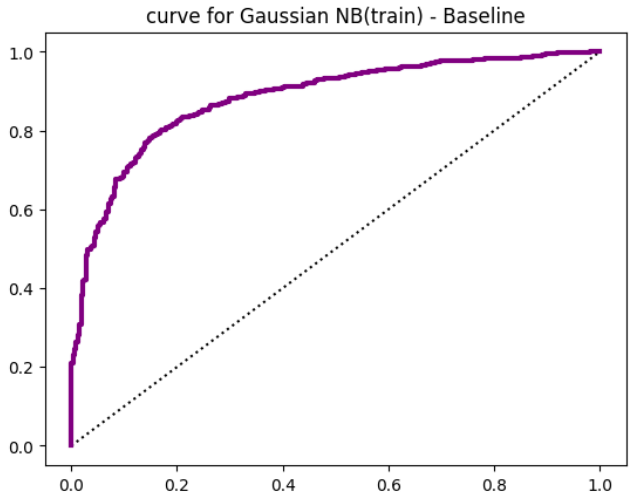
Gaussian Naïve Bayes

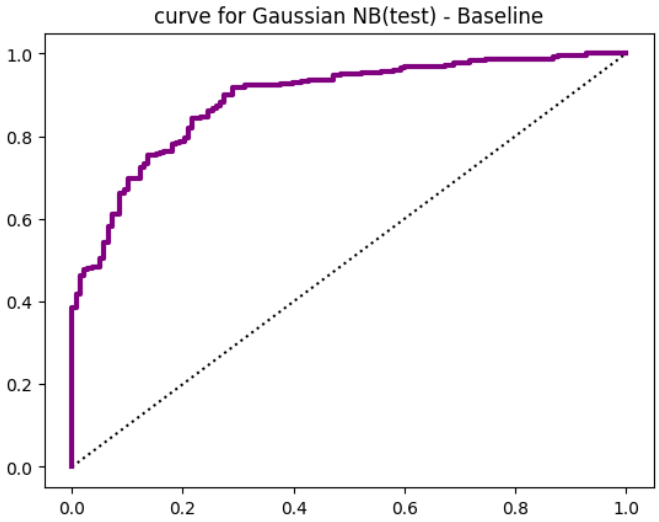
The below table is a comprehensive look of the models.

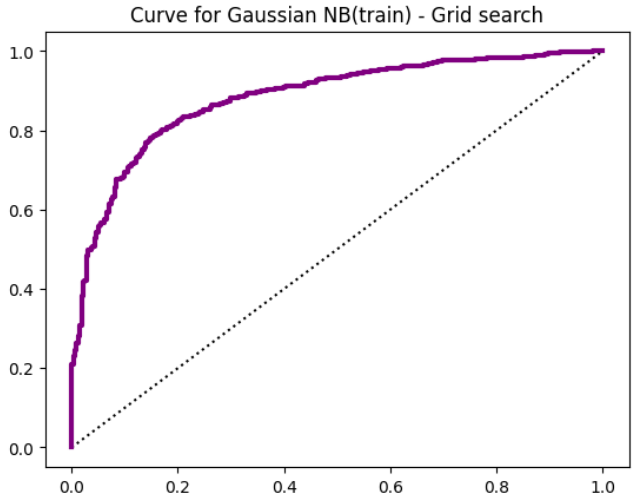
Model type	T/T	Accuracy	Precision		Recall		AUC score	True (-)	False (+)	False (-)	True (+)	F1 score	
			Class 0	Class 1	Class 0	Class 1						Class 0	Class 1
Baseline Model	Train	82.46	0.72	0.87	0.70	0.88	0.882	224	98	88	651	0.71	0.88
	Test	84.42	0.76	0.88	0.71	0.90	0.889	98	40	31	287	0.73	0.89
Grid_search Model	Train	82.46	0.72	0.87	0.70	0.88	0.882	224	98	88	651	0.71	0.88
	Test	84.42	0.76	0.88	0.71	0.90	0.889	98	40	31	287	0.73	0.89

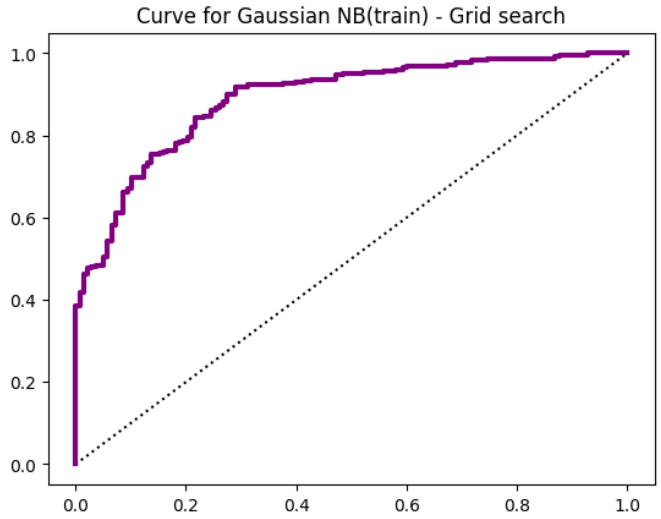
Both models have similar accuracy for the training (~82.46%) and test sets (~84.42%). This suggests that the models are generally correct about 82-84% of the time. they predict Class 0 or Class 1, they are correct 72-76% and 87-88% of the time, respectively. The F1 score for Class 0 is around 0.71-0.73, and for Class 1, it's around 0.88-0.89. This suggests the model is better at identifying Class 1 instances.

both models perform similarly across all metrics and data splits (training and testing), and there's no significant improvement achieved through grid search.









The above plots are very different, the train plots are very smooth compares to the test plots, even though the metrics of these models are similar numerically the plots provide another aspect of them. The models are able to identify class 1(i.e 'Conservative') better than class '0'(Labour)

With the test models, we can see that the initial values are moving orthogonal to the axis line, they are predicting in a very accurate manner, and then it goes haywire.

K-Nearest Neighbor

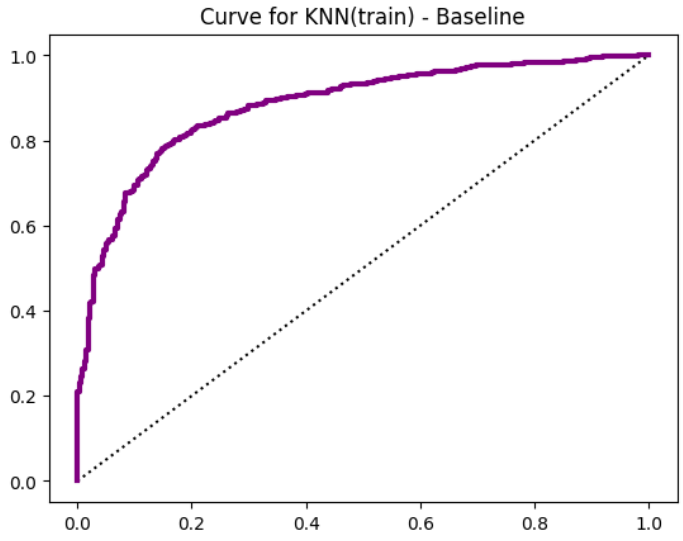
The below table is a comprehensive look of the models.

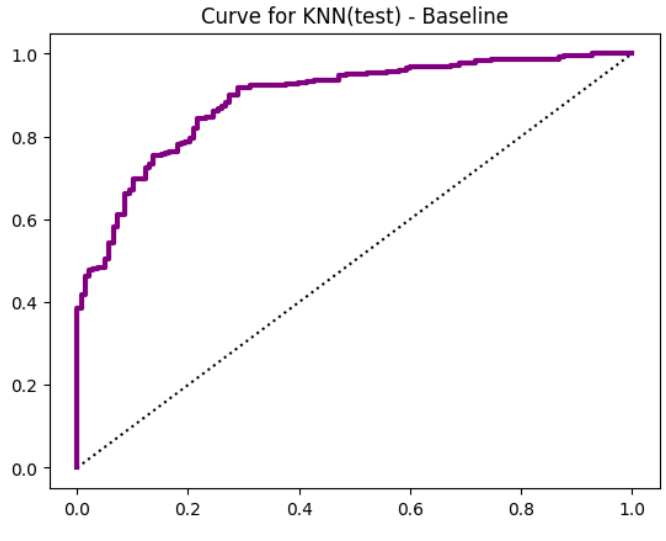
Model type	T/T	Accuracy	Precision		Recall		AUC score	True (-)	False (+)	False (-)	True (+)	F1 score	
			Class 0	Class 1	Class 0	Class 1						Class 0	Class 1
Baseline Model	Train	82.46	0.72	0.87	0.70	0.88	0.882	224	98	88	651	0.71	0.88
	Test	84.42	0.76	0.88	0.71	0.90	0.889	98	40	31	287	0.73	0.89
Grid_search Model	Train	82.46	0.72	0.87	0.70	0.88	0.882	224	98	88	651	0.71	0.88
	Test	84.42	0.76	0.88	0.71	0.90	0.889	98	40	31	287	0.73	0.89

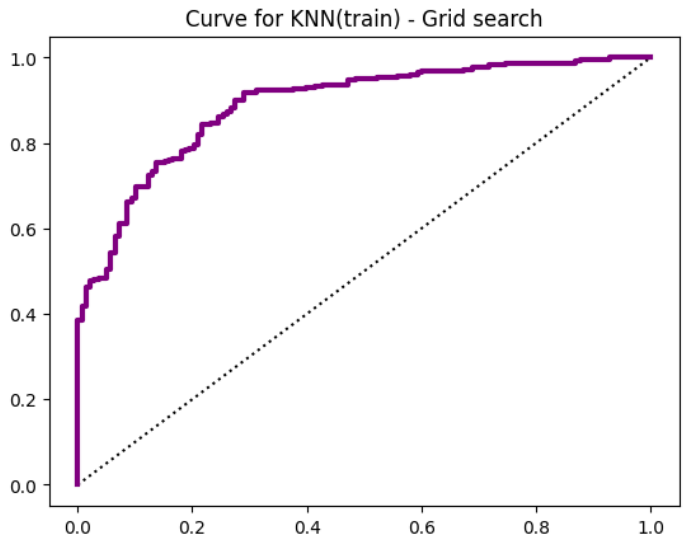
As discussed before, KNN model has generated the results which is an exact copy of what we foud in Gaussian NB. The scores are exactly the same.

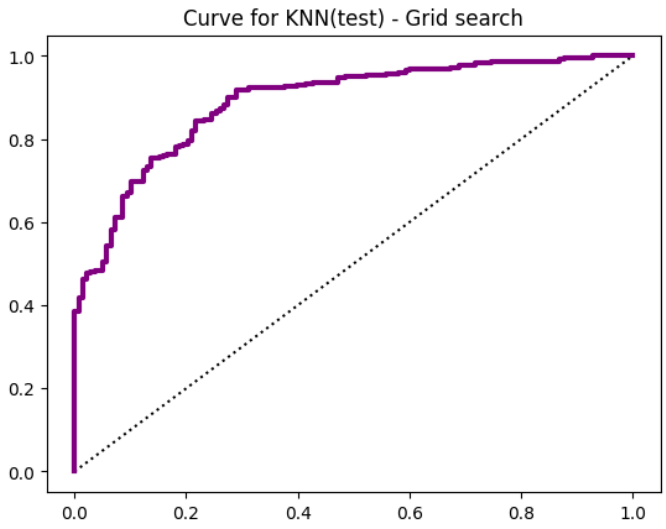
Both models have similar accuracy for the training (~82.46%) and test sets (~84.42%). This suggests that the models are generally correct about 82-84% of the time. they predict Class 0 or Class 1, they are correct 72-76% and 87-88% of the time, respectively. The F1 score for Class 0 is around 0.71-0.73, and for Class 1, it's around 0.88-0.89. This suggests the model is better at identifying Class 1 instances.

both models perform similarly across all metrics and data splits (training and testing), and there's no significant improvement achieved through grid search.









The train and test plots are identical and the same when compared between the baseline and grid search model. They are also indentical to the Gaussian NB model.

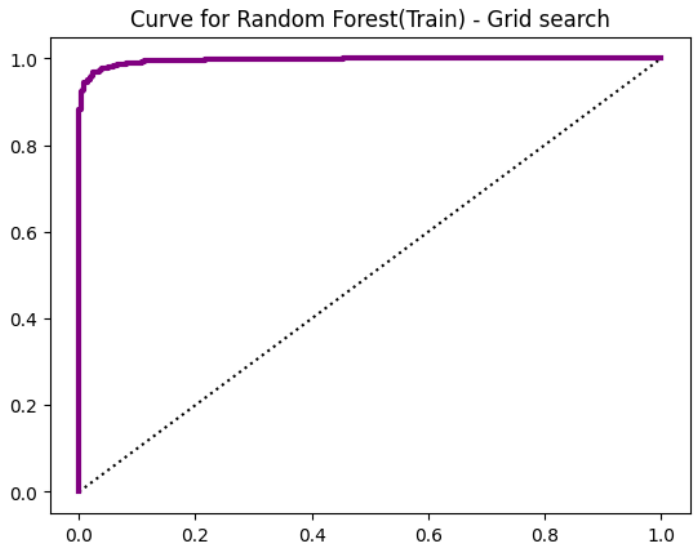
Random Forest Classifier

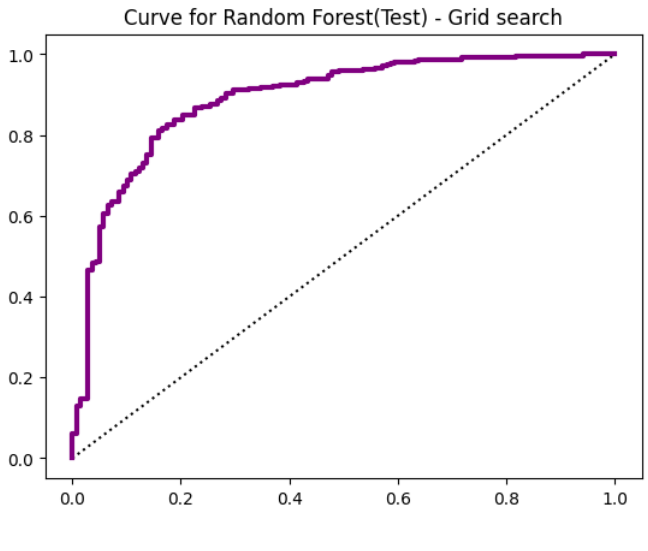
Model type	T/T	Accuracy	Precision		Recall		AUC score	True (-)	False (+)	False (-)	True (+)	F1 score	
			Class 0	Class 1	Class 0	Class 1						Class 0	Class 1
Grid_search Model	Train	96.88	0.96	0.97	0.94	0.98	0.996	303	19	14	725	0.95	0.98
	Test	83.55	0.75	0.87	0.68	0.90	0.888	94	44	31	287	0.71	0.88

The random forest classifier is an ensemble method, where we have a combination of different decision trees and their clubbed value is our classification output.

The significant drop in performance metrics from training to test set suggests the model is overfitting. It's performing exceptionally well on the training data but not as well on unseen data. Even though it is overfitting the AUC score remains quite high at 0.888, which is a positive sign that the model does perform well.

The model seems better at identifying Class 1 over Class 0, especially evident in the test set recall and F1 score.





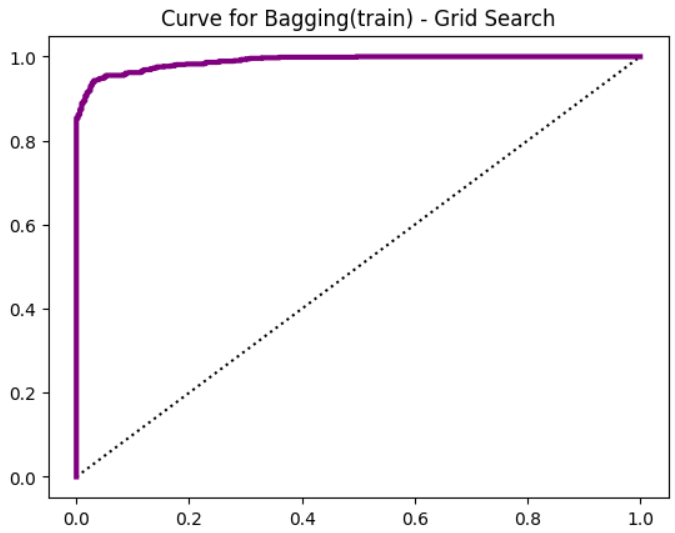
From the above plots we can see a very desirable AUC curve for the trainig dataset, this almost squarish plot puts the model at a high rating. But the corresponding test drops in performance, this is a sign of overfitting, meaning the model remembers train information and is quickly able to assign the correct class, while on unseen data it struggles.

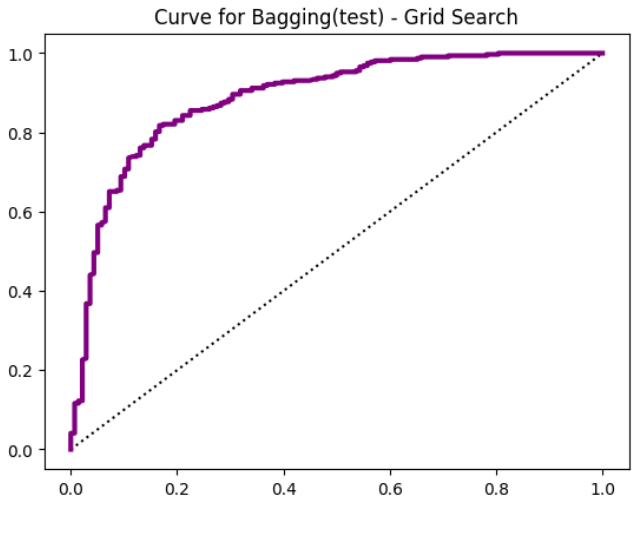
Bagging Classifier

Model type	T/T	Accuracy	Precision		Recall		AUC score	True (-)	False (+)	False (-)	True (+)	F1 score	
			Class 0	Class 1	Class 0	Class 1						Class 0	Class 1
Grid_search Model	Train	93.96	0.93	0.94	0.86	0.97	0.989	278	44	20	719	0.90	0.96
	Test	843.11	0.75	0.86	0.66	0.91	0.887	91	47	30	288	0.70	0.88

Similar to Random Forest, bagging is also overfitting the data as there is a drop in all the performance characteristics.

The AUC score still helps to maintain the model as a very good one. The train data becomes aspirational and the test data is the reality. Recall drops to 0.66 for class '0' but even still it retains the high value of 0.91 for class '1'. The decline in F1 score also shows a reduction of the balance between the classes.





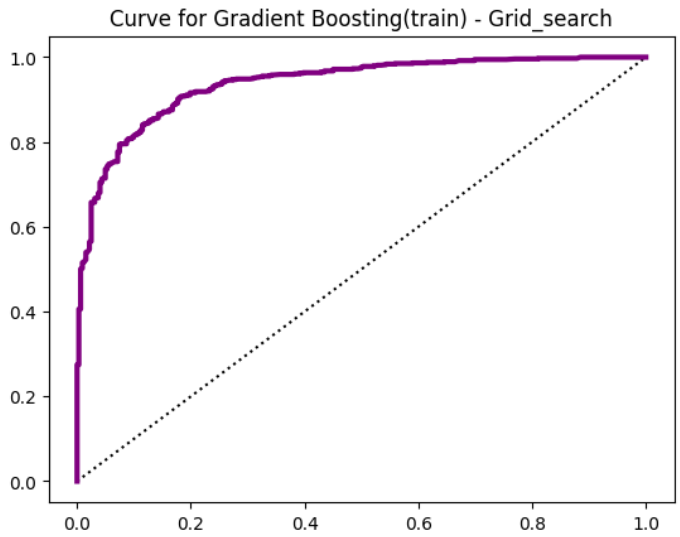
The AUC curve for the train model is the smoothest curve of the all the models we have tried now.

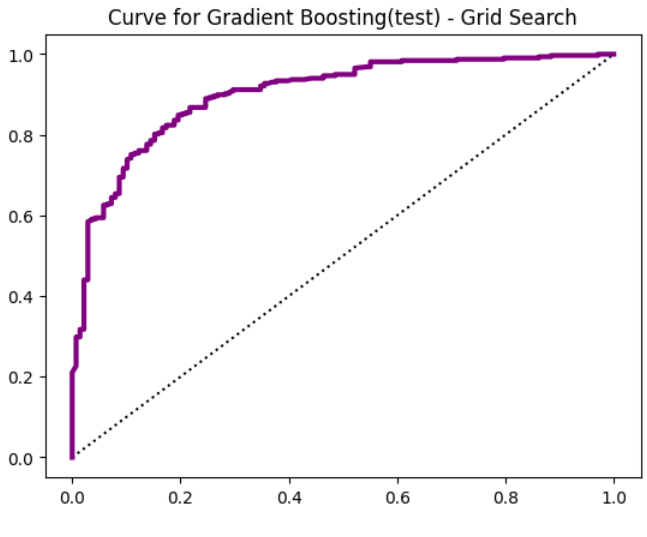
A smooth curve usually suggests that the model's performance is consistent across different thresholds. A smoother curve can often indicate better generalization, as it shows that the model performs similarly across a range of decision thresholds. But the same cannot be said about the test curve, it is jagged and rough. Meaning the model is not able to perform at all thresholds.

Gradient Boosting Classifier

Model type	T/T	Accuracy	Precision		Recall		AUC score	True (-)	False (+)	False (-)	True (+)	F1 score	
			Class 0	Class 1	Class 0	Class 1						Class 0	Class 1
Grid_search Model	Train	88.12	0.85	0.89	0.75	0.94	0.937	240	82	44	695	0.79	0.92
	Test	83.77	0.77	0.86	0.67	0.91	0.901	92	46	28	290	0.71	0.89

With an accuracy of 88.12%, the model seems to be doing a good job on the training set, but the accuracy drops a bit to 83.77% on the test set.The model is accurate when it predicts either class, courtesy of precision. The model is excellent at capturing class 1 but lacks it in class 0. The AUC score of 0.937 is excellent even for the test the score remains at 0.901, suggesting that the model is good at distinguishing between the two classes. The F1 score is also balanced overall. There's a small drop in performance metrics from the training set to the test set, but it's not drastic, suggesting the model is neither overfitting nor underfitting severely.



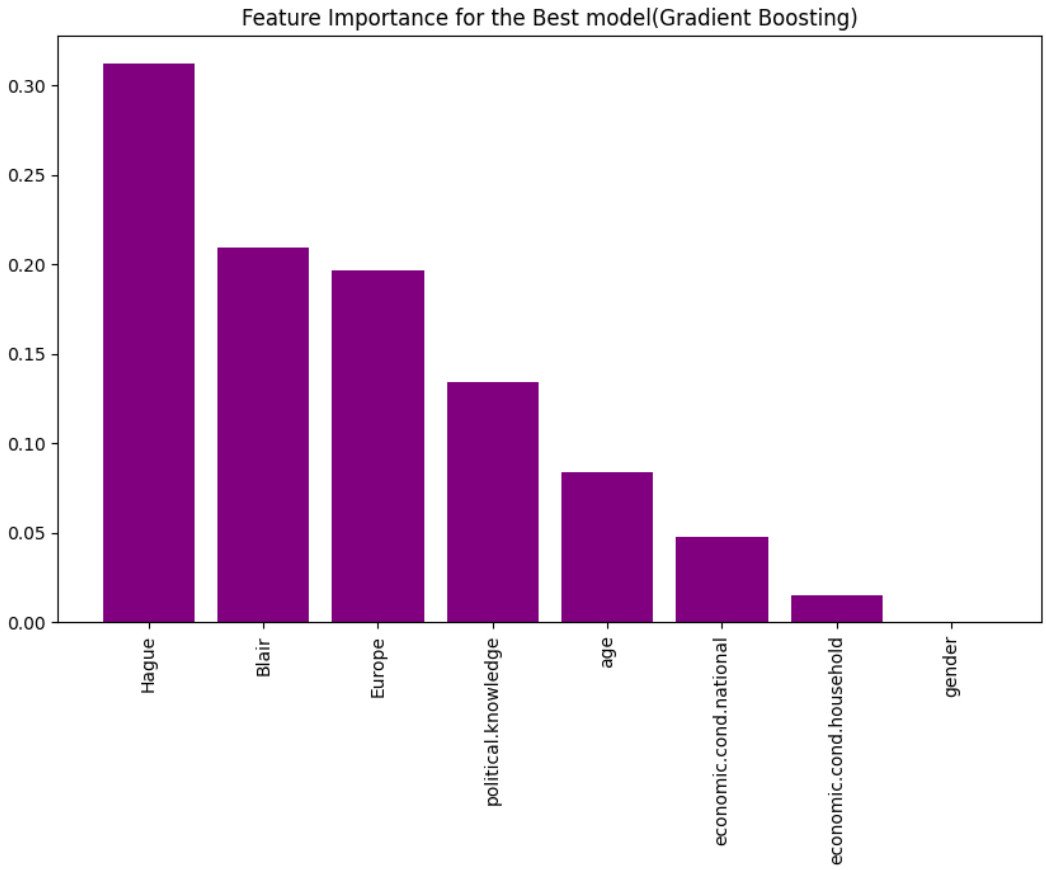


The curves are moderately jagged not as smooth as Bagging classifier, yet the performance of this model is better than others.

The overall most suited model for this data is the gradient boost classifier model. This performs admirably in both the test and train data.

1.8) 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.

Please find the below feature importances of the best performing model.



Insights:

- "Hague" (Conservative leader) and "Blair" (Labour leader) have high importance scores. This implies that public opinion of these leaders is a strong indicator of voting behavior.
- "Europe" is the next major factor, indicating that sentiments about European integration may be a decisive factor.
- "political.knowlegde", a non-negligible importance score suggests that voters who are aware of party positions, on European integration, are more carefull to whom they cast their vote.
- National and household economic conditions seem to have lower importance. Surprisingly, household economics impact voting less than national economics.
- "gender" has the least importance, this is a positive sign socially as out society attemtps to break the sex based pre-conceptions and ancient view.

Recommendations:

- Given the high importance of "Hague" and "Blair," CNBE should consider keeping a close eye on the actions of either side representatives. Segments that specifically analyze these leaders whereabouts throughout the election day.
- The channel could run a poll of some sort that captures the eurcentric sentiments from various social platforms every hour. This would add another layer of depth to the exit poll analysis.
- Given the importance of political knowledge on the european integreation front, constant updates and intervies diving into the mind of the voter from multiple polling stations by the channel could see a viewable mark on the exit polls predictions.
- Even though economic conditions are not the most important factors, they still play a role. CNBC should consider a segment summarizing the current national and household economic conditions could provide a comprehensive view. This could be what the representative have promised and the current on the ground situations.
- The channel should remove sex(male and female) based questions and related updates as they do not make a significant impact. They can be wholly scrapped off.

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:

- President Franklin D. Roosevelt in 1941
- President John F. Kennedy in 1961
- 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 Roosevelt's speech from 1941

"On each national day of inauguration since 1789, the people have renewed their sense of dedication to the United States.\n\nIn Washington\'s day the task of the people was to create and weld together a nation.\n\nIn Lincoln\'s day the task of the people was to preserve that Nation from disruption from within.\n\nIn this day the task of the people is to save that Nation and its institutions from disruption from without.\n\nTo us there has come a time"

Sample of Kennedy's Speech from 1961

"Vice President Johnson, Mr. Speaker, Mr. Chief Justice, President Eisenhower, Vice President Nixon, President Truman, reverend clergy, fellow citizens, we observe today not a victory of party, but a celebration of freedom -- symbolizing an end, as well as a beginning -- signifying renewal, as well as change. For I have sworn I before you and Almighty God the same solemn oath our forebears l prescribed nearly a century and three quarters ago.\n\nThe world is very different now. For man holds in his mortal hands the power to abolish all forms of human poverty and all forms of human life. And yet the same revolutionary beliefs for which our forebears fought are still at issue around the globe".

Sample of Nixon's Speech from 1973

"Mr. Vice President, Mr. Speaker, Mr. Chief Justice, Senator Cook, Mrs. Eisenhower, and my fellow citizens of this great and good country we share together:\n\nWhen we met here four years ago, America was bleak in spirit, depressed by the prospect of seemingly endless war abroad and of destructive conflict at home.\n\nAs we meet here today, we stand on the threshold of a new era of peace in the world.\n\nThe central question before us is: How shall we use that peace? Let us resolve that this era we are about to enter will not be what other postwar periods have so often been: a time of retreat and isolation that leads to stagnation at home and invites new danger abroad".

The table below shows the Characters, words and sentences that are present within each presidential speech.

Speech Name	Characters	Words	Sentences
Roosevelt 1941	7571	1536	68
Kennedy 1961	7618	1546	52
Nixon 1973	9991	2028	69

Nixon's 1973 speech has the most characters, the most words and also the most sentences. Kenenedy's 1961 speech takes the second spot in terms of character count and word count, but it has the lowest number of sentences out of the three.

Roosevelt's 1941 speech has the least amount of characters, and the least amount of words. it is marginally smaller than Kennedy's speech, but it is has the second largest amount of sentences.

Roosevelt's speech has more sentences, though having the lowest character and word count, which is the opposite for Kennedy's speech. Nixon's speech has the most number of words and characters but also has only one sentence more than Roosevelt's. This means that Nixon's speech has more words per sentence on average.

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.

Stop words are a set of commonly used words in a language that are filtered out before or after processing of natural language data (text) because they are insignificant. There is no single universal list of stop words, nor any agreed upon rules for identifying stop words. but we still so have list that is very commonly occuring. Therefore, any group of words can be chosen as the stop words for a given purpose.

Examples of stopwords: These are some of the most occuring

- a
- an
- the
- and
- or
- but

we will use the available stopwords library from NLTK, we can add words that we consider as stopwords. Hence it is totally upto out discretion and case by case proceedings.

While removing the stopwords we will also remove the punctuations and other symbols which do not convey huge information.

Stopwords from Roosevelt's Speech of 1941

- Words before removing stopwords = 1536
- Words after removing stopwords = 627

We can see that almost more than half of the words from Roosevelt's speech was only stopwords.

Please find the sample of the speech after removing stopwords.

['national', 'day', 'inauguration', 'since', 'people', 'renewed', 'sense', 'dedication', 'united', 'states', 'washington', 'day', 'task']

Stopwords from Kennedy's Speech of 1961

- Words before removing stopwords = 1546
- Words after removing stopwords = 692

We can see that almost more than half of the words from Kennedy's speech was only stopwords. Similar to Roosevelt's.

Please find the sample of the speech after removing stopwords.

['vice', 'president', 'johnson', 'mr', 'speaker', 'mr', 'chief', 'justice', 'president', 'eisenhower', 'vice', 'president', 'nixon', 'president']

Stopwords from Nixon's Speech of 1973

- Words before removing stopwords = 2028
- Words after removing stopwords = 832

We can see that almost 2/3rds of the speech from Nixon only consistest of stopwords, which is greater than those of Kennedy's and Roosevelt's.

Please find the sample of the speech after removing stopwords.

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

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)

Roosevelt's speech from 1943

These are Roosevelt's most spoken words from his speech

Word	Count
Nation	12
know	10
spirit	9

The word "Nation" has occurred 12 times which is almost 2% of the total words spoken words after removing stowords and punctuation.

Kennedy's speech from 1961

These are Kennedy's most spoken words from his speech

Word	Count
Let	16
Us	12
World	8

The word "Let" has occurred 16 from a total count of 692(after removing the stopwords and punctuation). This alone has taken 2.3% of the total word count volumn.

Nixon's speech from 1973

These are Nixon's most spoken words from his speech

Word	Count
Us	26
Let	22
america	21

The word "Us" which was the second most occuring word from Kennedy's speech has taken the top spot in Nixon's speech. It has occurred 26 from a total count of 832(after removing the stopwords and punctuation). This alone has taken 3.1% of the total word count volumn.

Nixon's speech is also the one where the top three word all cross the 20 counts mark. All other speech's had atleast one word with a single digit value.

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

Word cloud of Roosevelt's Speech

Darker the shade and bigger words represent more important words within the text given, in this case it would be the speech made by Roosevelt in the year 1941.

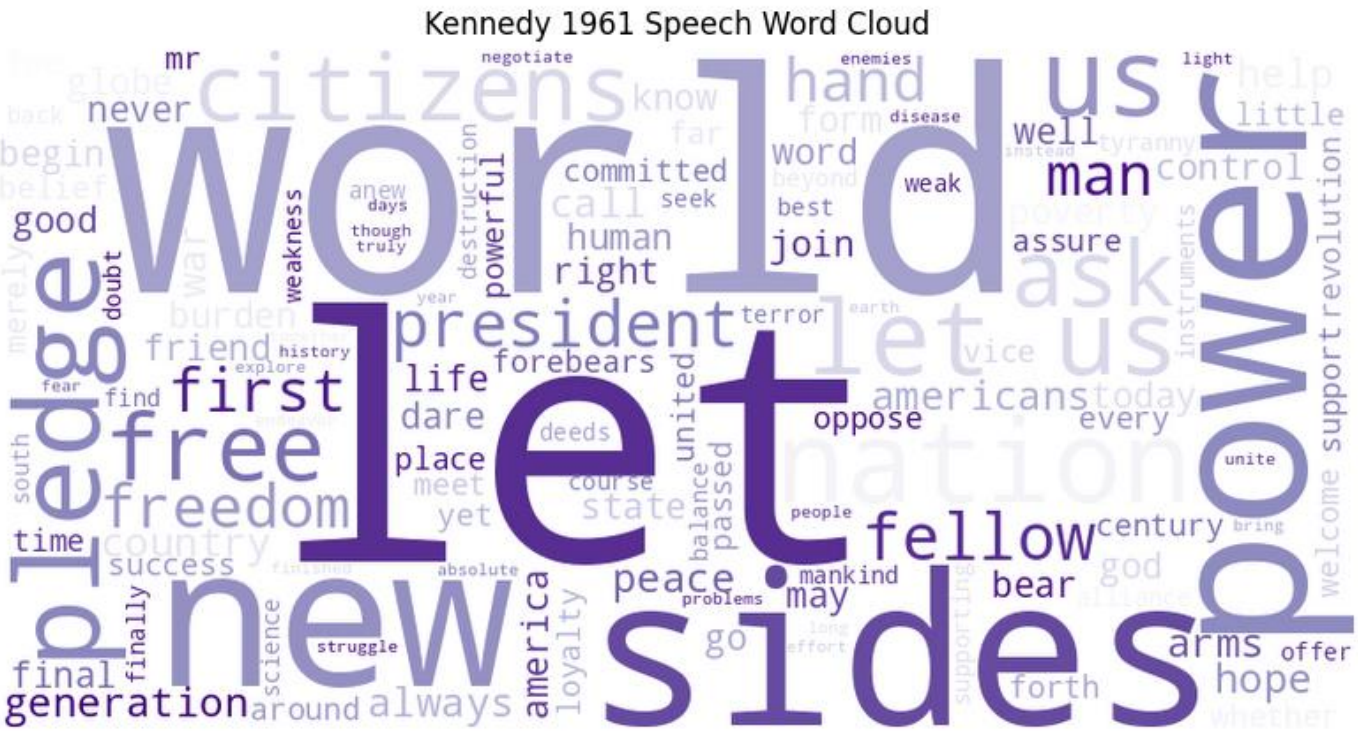
"Nation" and "Know", these words are bigger, suggesting they are more frequent or more important in the context of your dataset. **"Year"** being smaller indicates it's less frequent but still relevant.

Lighter Shade (Nation, Know, Year) might indicate a less intense focus or lower frequency compared to darker shades. **Darker Shade (Spirit, Freedom, New, Life)** could signify stronger emphasis or higher frequency.

The total word count for Roosevelt's speech was 627 and we can only see a small portion in the word cloud, thereby showcasing its value.

We know these are speeches from presidents hence they would be sociopolitical and socioeconomic themed, and the words project the same.

Word cloud of Kennedy's speech



Darker the shade and bigger words represent more important words within the text given, in this case it would be the speech made by Kennedy in the year 1961.

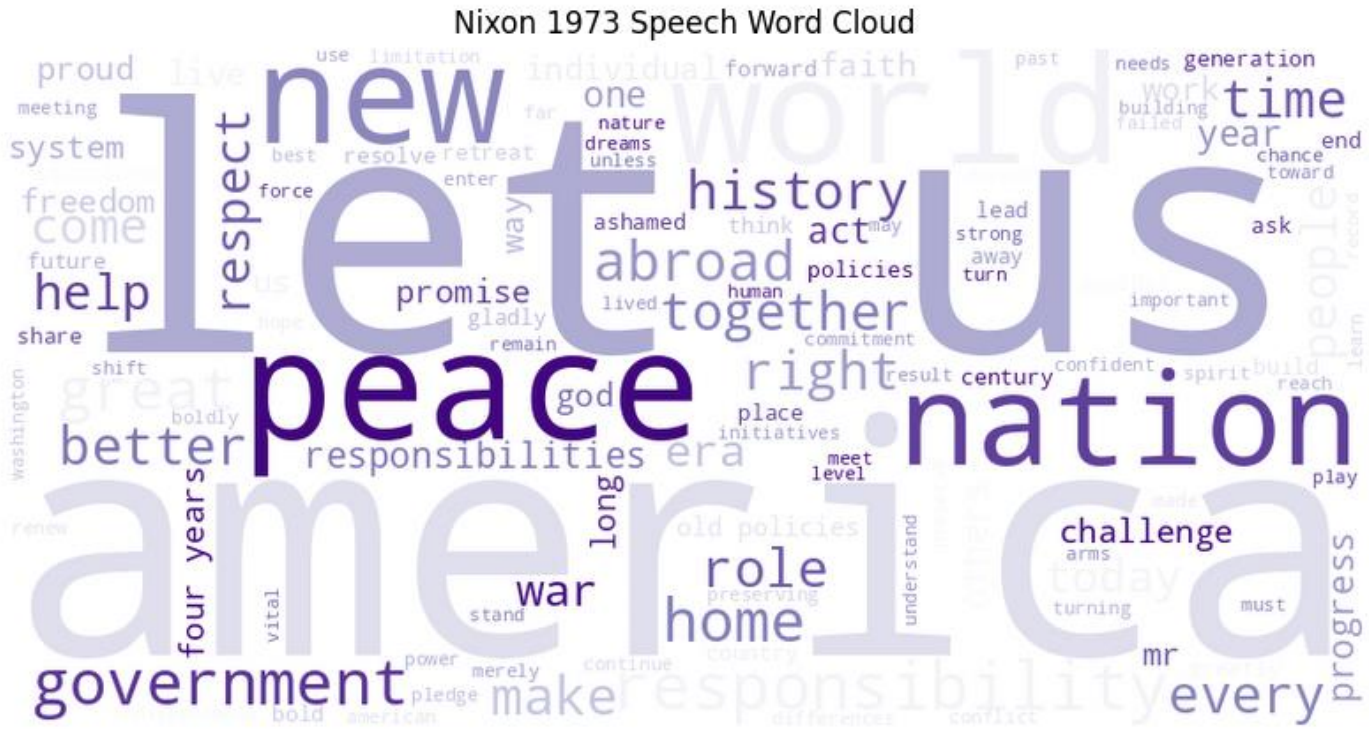
"World", "Power" and "pledge" , these words are bigger and moderately sized, suggesting they are more frequent or more important in the context of your dataset. **"ask"** being smaller indicates it's less frequent but still relevant.

Lighter Shade (citizens, let, revolution) might indicate a less intense focus or lower frequency compared to darker shades. **Darker Shade (generation, first, bear)** could signify stronger emphasis or higher frequency.

The total word count for Kennedy's speech was 692(post stop words removal), the image hosts only a select portion, showing their importance.

We know these are speeches from presidents hence they would be sociopolitical and socioeconomic themed, and the words project the same.

Word cloud of Nixon's speech



Darker the shade and bigger words represent more important words within the text given, in this case it would be the speech made by Nixon in the year 1973.

As detailed in other word clouds, the size of the words could determine frequency and related relevance. The shade of the word represents emphasis. The total word count for Nixon's speech after removing stop words was 832, in the word cloud we can only see just a small portion of it. These are important words in some sense hence they are present in this visualization.

We know these are speeches from presidents hence they would be sociopolitical and socioeconomic themed, and the words project the same.