Executive Summary:-

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

Introduction: -

The purpose of this whole exercise is to explore the dataset. Do the exploratory data analysis. Explore the dataset and analysis of using method all seven modal. To create an exit poll that will help in predicting overall win and seats covered by a particular party.

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

Data Description:-

Data Dictionary	
Vote	Party choice: Conservative or Labour
Age	In years
Economic.cond.national	Assessment of current national economic conditions, 1 to 5
Economic.cond.household	Assessment of current household economic conditions, 1 to 5.
Blair	Assessment of the Labour leader, 1 to 5.
Hague	Assessment of the Conservative leader, 1 to 5.
Europe	An 11-point scale that measures respondents' attitudes toward European integration. High scores represent 'Eurosceptic' sentiment.
Political.knowledge	Knowledge of parties' positions on European integration, 0 to 3.
Gender	Female or male.

Tab-1.a

Sample of the Dataset:-

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

Fig-1.0

Conclusion | insight: This is Sample of given data all information is mention as per exit poll that will help in predicting overall win and seats covered by a particular party .

Data Describe:-

	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	1525.0	763.000000	440.373894	1.0	382.0	763.0	1144.0	1525.0
age	1525.0	54.182295	15.711209	24.0	41.0	53.0	67.0	93.0
economic.cond.national	1525.0	3.245902	0.880969	1.0	3.0	3.0	4.0	5.0
economic.cond.household	1525.0	3.140328	0.929951	1.0	3.0	3.0	4.0	5.0
Blair	1525.0	3.334426	1.174824	1.0	2.0	4.0	4.0	5.0
Hague	1525.0	2.746885	1.230703	1.0	2.0	2.0	4.0	5.0
Europe	1525.0	6.728525	3.297538	1.0	4.0	6.0	10.0	11.0
political.knowledge	1525.0	1.542295	1.083315	0.0	0.0	2.0	2.0	3.0

Fig-1.1

Conclusion | insight: Data min value of data 0.0 and max value is 93 and other thing zero is available in data. Required scaling for only KNN Modal due to data range is only one column higher.

Exploratory data analysis:-

```
RangeIndex: 1525 entries, 0 to 1524
Data columns (total 10 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Unnamed: 0              1525 non-null   int64
 1   vote                    1525 non-null   object
 2   age                     1525 non-null   int64
 3   economic.cond.national  1525 non-null   int64
 4   economic.cond.household 1525 non-null   int64
 5   Blair                   1525 non-null   int64
 6   Hague                   1525 non-null   int64
 7   Europe                  1525 non-null   int64
 8   political.knowledge     1525 non-null   int64
 9   gender                  1525 non-null   object
dtypes: int64(8), object(2)
memory usage: 119.3+ KB
```

Fig-1.1.0

Conclusion | insight: Ten types of information given in data as a column. Data is memory 119.3+ MB and 0 to 1524 row and 10 columns.

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

Bivariate Analysis

Distribution plots:-

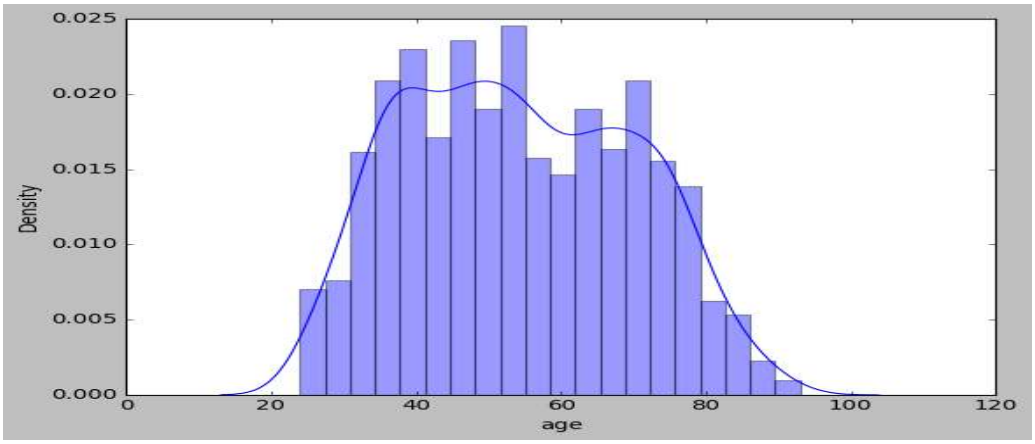

Fig-1.2

Countplot:-

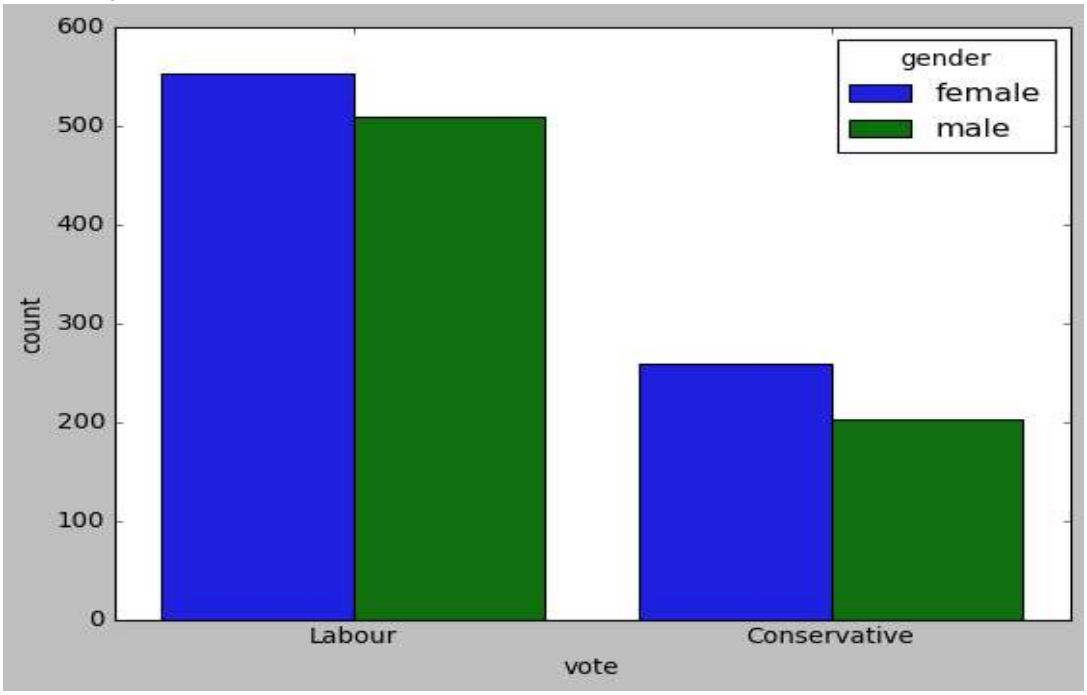

Fig-1.3

Conclusion | insight: Distribution and count plot are performed Vote category and Age are also show in range.

Pair plot:-

Fig-1.4

Pair plot:-

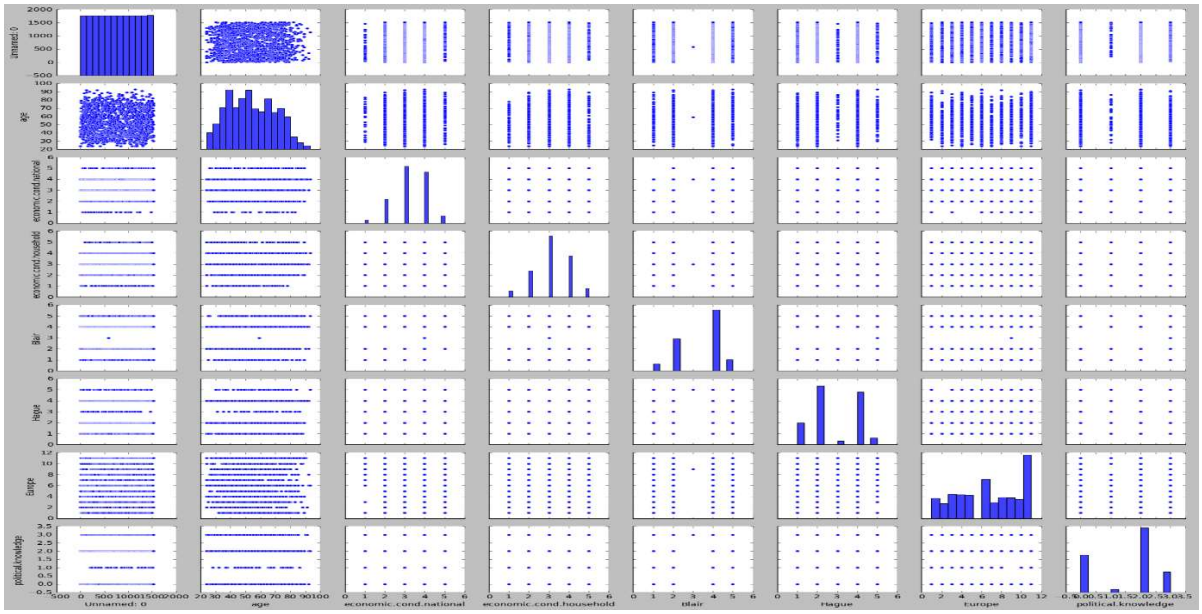

Fig-1.5

Conclusion | insight: Pair plot are show in both fig-1.4 & 1.5 behave of data are all column & Row.

Boxplot:-

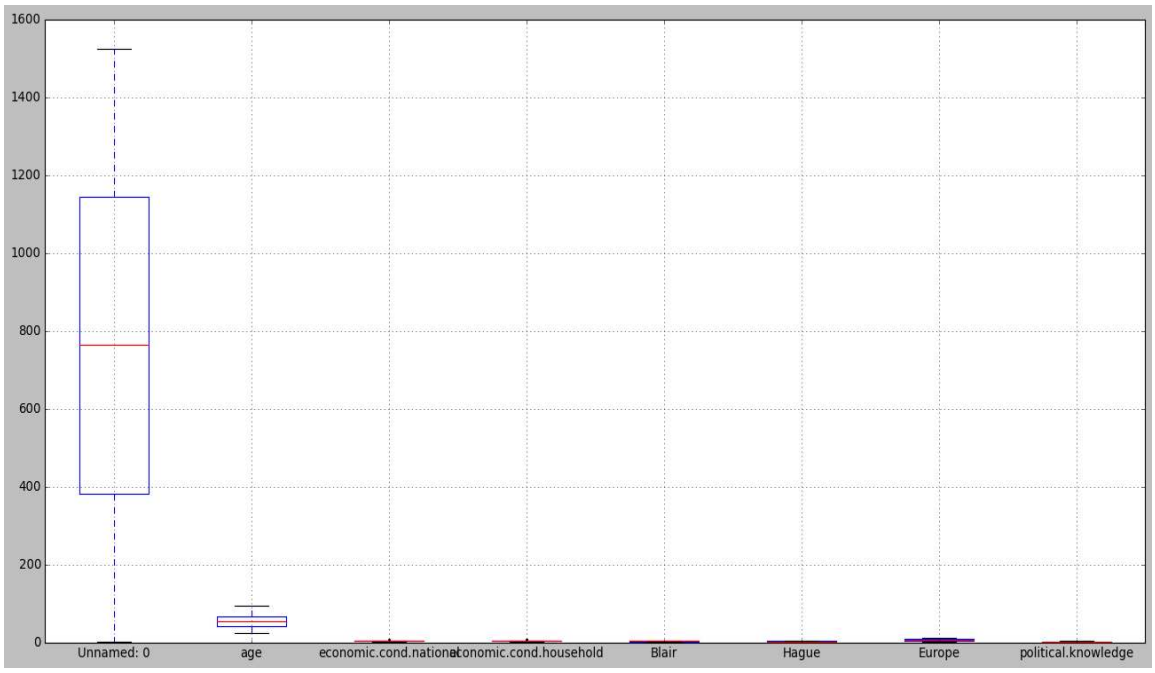

Fig-1.6

Conclusion | insight: After find outlier we are find two column are mention some outlier, column is economic.cond.natonatonomic & economic.cond.household both.

Correlation plots:-

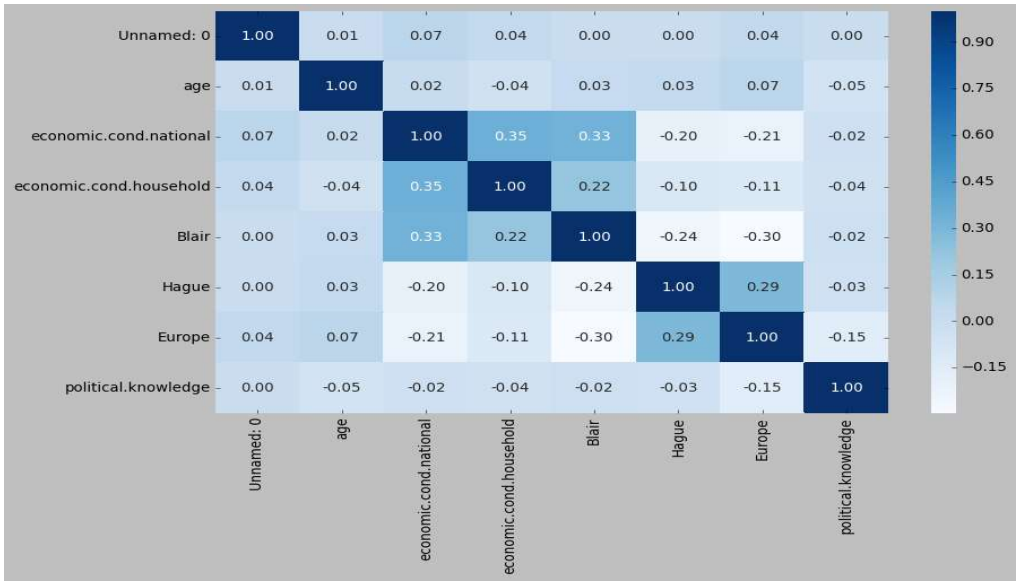

Fig-1.6

Conclusion | insight: we shown in fig-1.6 correlations of all column and showing some number to strong represents of inform in No.

Univariate Analysis

Sub Plots / Dist. Plots:-

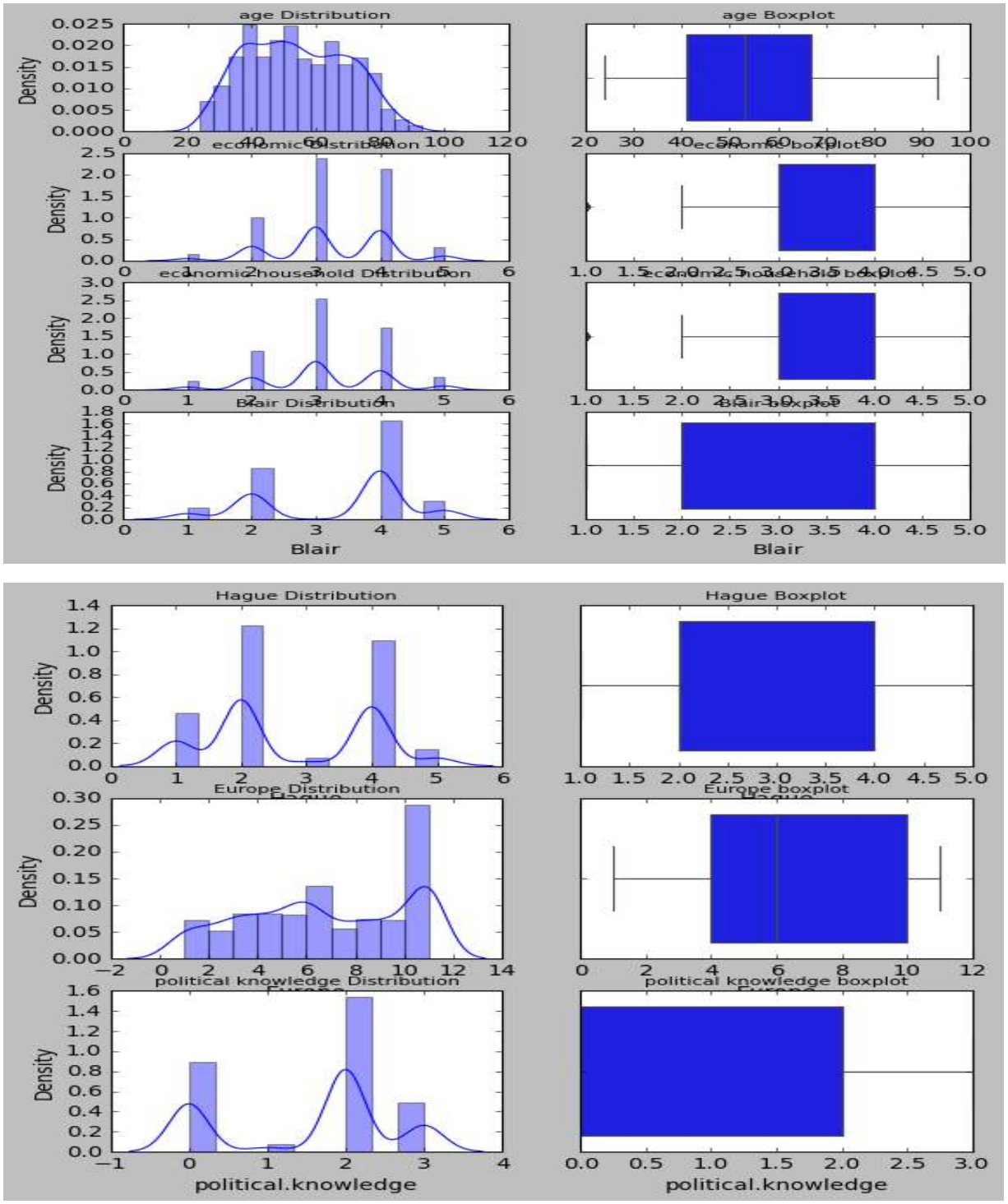

Fig-1.7

Conclusion | insight: Campier both modal as boxplot & distplot both modal as same time show what is data.

Q 1.3 Encode the data (having string values) for Modelling. Is Scaling necessary here or not? Data Split: Split the data into train and test.

Convert to Categorical:-

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

Fig-1.8

Conclusion | insight: Categorical format convert some data they have format in variable form to convert numerical format for use all & ML modal also.

Duplicate Data sample:-

	Unnamed: 0	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	IsLabour_or_not	IsMale_or_not
0	1	43	3	3	4	1	2	2	1	0
1	2	36	4	4	4	4	5	2	1	1
2	3	35	4	4	5	2	3	2	1	1
3	4	24	4	2	2	1	4	0	1	0
4	5	41	2	2	1	1	6	2	1	1
...	...	...	...	...	...	...	...	...	...	...
1520	1521	67	5	3	2	4	11	3	0	1
1521	1522	73	2	2	4	4	8	2	0	1
1522	1523	37	3	3	5	4	2	2	1	1
1523	1524	61	3	3	1	4	11	2	0	1
1524	1525	74	2	3	2	4	11	0	0	0

Fig-1.9

Conclusion | insight: No any Duplicate data there in data set.

Data Describe:-

	count	mean	std	min	25%	50%	75%	max
Unnamed: 0	1525.0	763.000000	440.373894	1.0	382.0	763.0	1144.0	1525.0
age	1525.0	54.182295	15.711209	24.0	41.0	53.0	67.0	93.0
economic.cond.national	1525.0	3.245902	0.880969	1.0	3.0	3.0	4.0	5.0
economic.cond.household	1525.0	3.140328	0.929951	1.0	3.0	3.0	4.0	5.0
Blair	1525.0	3.334426	1.174824	1.0	2.0	4.0	4.0	5.0
Hague	1525.0	2.746885	1.230703	1.0	2.0	2.0	4.0	5.0
Europe	1525.0	6.728525	3.297538	1.0	4.0	6.0	10.0	11.0
political.knowledge	1525.0	1.542295	1.083315	0.0	0.0	2.0	2.0	3.0
IsLabour_or_not	1525.0	0.697049	0.459685	0.0	0.0	1.0	1.0	1.0
IsMale_or_not	1525.0	0.467541	0.499109	0.0	0.0	0.0	1.0	1.0

Fig-1.10

Conclusion | insight: All data describe in some parameter so we now that behave of data and future action for perform operation.

Modelling Is Scaling Necessary: -

Almost data has similar, only one column Age is quietly different as for rest modal, using just like KNN modal for best result using scaling otherwise rest modal not need to scaling. We are not going to scale the data for Logistic regression, LDA and Naive Baye's models as it is not necessary.

Splitting the Train-Test Data:-

```
from sklearn.model_selection import train_test_split
```

Split the data into train and test

```
X=df.drop('IsLabour_or_not',axis=1)
Y=df['IsLabour_or_not']
```

```
X_train,X_test, Y_train, Y_test=train_test_split(X,Y,train_size=0.70, random_state=1)
```

Fig-1.11

Conclusion | insight: Splitting data in two part train in 70% and 30% test .in this case "Isolabour_or_not" are using for target data .all information are nearby this "Isolabour_or_not" column .

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

Logistic Regression -LR train

The Predicted Classes and Probs- Logistic_Model:-

	0	1
0	0.629796	0.370204
1	0.180228	0.819772
2	0.191912	0.808088
3	0.169680	0.830320
4	0.054035	0.945965

Fig-1.12

Confusion Matrix:-

```
[229, 103]
[ 70, 665]
```

Fig-1.13

Accuracy & Classification: -

```
Logistic_train_acc=Logistic_model.score(X_train,Y_train)
print(Logistic_train_acc)
```

```
0.837863167760075
```

```
print(classification_report(Y_train,ytrain_predict_LR1))
```

```
              precision    recall  f1-score   support

           0       0.77      0.69      0.73       332
           1       0.87      0.90      0.88       735

    accuracy                           0.84      1067
   macro avg       0.82      0.80      0.81      1067
weighted avg       0.83      0.84      0.84      1067
```

Fig-1.14

Conclusion | insight: All are parameter just like accuracy and confusion matrix in LR modal Train data result was shown in fig-1.13 & 1.14.

Logistic Regression LR test

The Predicted Classes and Probs - Logistic_Model:-

	0	1
0	0.930667	0.069333
1	0.696081	0.303919
2	0.323824	0.676176
3	0.475360	0.524640
4	0.150966	0.849034

Fig-1.15

Confusion Matrix:-

```
[ 85,  45],
[ 36, 292]
```

Fig-1.16

Accuracy & Classification: -

```
Logistic_test_acc=Logistic_model.score(X_test,Y_test)
print(Logistic_test_acc)
```

```
0.8231441048034934
```

```
print(classification_report(Y_test,ytest_predict_LR1))
```

```
              precision    recall  f1-score   support

           0       0.70      0.65      0.68       130
           1       0.87      0.89      0.88       328

    accuracy                           0.82       458
   macro avg       0.78      0.77      0.78       458
weighted avg       0.82      0.82      0.82       458
```

Fig-1.17

Conclusion | insight: All are parameter just like accuracy and confusion matrix in LR modal Test data result was shown in fig-1.16 & 1.17.

LDA (linear discriminant analysis)

LDA Train:-

The Predicted Classes and Probs - LDA_Model:-

	0	1
0	0.655725	0.344275
1	0.156789	0.843211
2	0.186375	0.813625
3	0.136064	0.863936
4	0.040668	0.959332

Fig-1.18

Confusion Matrix:-

```
[235,  97]
[ 75, 660]
```

Fig-1.19

Accuracy & Classification: -

```
LDA1_train_acc=LDA_model.score(X_train,Y_train)
print(LDA1_train_acc)
```

```
0.8388003748828491
```

```
print(classification_report(Y_train,ytrain_predict_LDA1 ))
```

```
              precision    recall  f1-score   support

           0       0.76      0.71      0.73       332
           1       0.87      0.90      0.88       735

    accuracy                           0.84      1067
   macro avg       0.81      0.80      0.81      1067
weighted avg       0.84      0.84      0.84      1067
```

Fig-1.20

Conclusion | insight: All are parameter just like accuracy and confusion matrix in LDA modal Train data result was shown in fig-1.19 & 1.20.

LDA Test:-

The Predicted Classes and Probs - LDA_Model:-

	0	1
0	0.950899	0.049101
1	0.745815	0.254185
2	0.321758	0.678242
3	0.484760	0.515240
4	0.135276	0.864724

Fig-1.21

Confusion Matrix:-

```
[ 86,  44],
[ 39, 289]
```

Fig-1.22

Accuracy & Classification: -

```
LDA1_test_acc=LDA_model.score(X_test,Y_test)
print(LDA1_test_acc)
```

```
0.8187772925764192
```

```
print(classification_report(Y_test,ytest_predict_LDA1 ))
```

```
              precision    recall  f1-score   support

           0       0.69      0.66      0.67       130
           1       0.87      0.88      0.87       328

    accuracy                           0.82       458
   macro avg       0.78      0.77      0.77       458
weighted avg       0.82      0.82      0.82       458
```

Fig-1.23

Conclusion | insight: All are parameter just like accuracy and confusion matrix in LDA modal Train data result was shown in fig-1.22 & 1.23.

Result of both Modal

1. Both the Model performs well on both training and test data.
2. There is no case of over fitting or under fitting for both the model.
3. **Scores of Logistic Regression model is marginally better than Linear Discriminant model.**

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

Scaling Train & Test:-

	Unnamed: 0	age	economic.cond.national	economic.cond.household	Blair	Hague	Europe	political.knowledge	IsMale_or_not
0	1	-0.711973	-0.279218	-0.150948	0.566716	-1.419886	-1.434426	0.422643	-0.937059
1	2	-1.157661	0.856268	0.924730	0.566716	1.018544	-0.524358	0.422643	1.067169
2	3	-1.221331	0.856268	0.924730	1.418187	-0.607076	-1.131070	0.422643	1.067169
3	4	-1.921698	0.856268	-1.226625	-1.136225	-1.419886	-0.827714	-1.424148	-0.937059
4	5	-0.839313	-1.414704	-1.226625	-1.987695	-1.419886	-0.221002	0.422643	1.067169

Fig-1.24

Conclusion | insight: Before building the KNN model, we need to scale the data as KNN is a distance based algorithm and Scaling will give similar weightage to all the variables.

KNN_Train:-

The Predicted Classes and Probs - KNN_Model:-

	0	1
0	0.2	0.8
1	0.4	0.6
2	0.2	0.8
3	0.4	0.6
4	0.6	0.4

Fig-1.25

Confusion Matrix:-

```
[165, 167]
[142, 593]
```

Fig-1.26

Accuracy & Classification: -

```
KNN1_train_acc=KNN_model.score(X_train,Y_train)
print(KNN1_train_acc)
```

```
0.7104029990627929
```

```
print(classification_report(Y_train,ytrain_predict_KNN1))
```

```
              precision    recall  f1-score   support

           0       0.54      0.50      0.52       332
           1       0.78      0.81      0.79       735

    accuracy                           0.71      1067
   macro avg       0.66      0.65      0.65      1067
weighted avg       0.70      0.71      0.71      1067
```

Fig-1.27

KNN Model Test:-

The Predicted Classes and Probs - KNN_Model:-

	0	1
0	0.4	0.6
1	0.4	0.6
2	0.8	0.2
3	0.4	0.6
4	0.2	0.8

Fig-1.28

Confusion Matrix:-

```
[ 46,  84]
[ 81, 247]
```

Fig-1.29

Accuracy & Classification: -

```
KNN1_test_acc=KNN_model.score(X_test,Y_test)
print(KNN1_test_acc)
```

```
0.6397379912663755
```

```
print(classification_report(Y_test,ytest_predict_KNN1 ))
```

```
              precision    recall  f1-score   support

           0       0.36      0.35      0.36       130
           1       0.75      0.75      0.75       328

    accuracy                           0.64       458
   macro avg       0.55      0.55      0.55       458
weighted avg       0.64      0.64      0.64       458
```

Fig-1.30

Naïve Bayes Model:-

Naïve Bayes Train-

The Predicted Classes and Probs - NB_Model:-

	0	1
0	0.674324	0.325676
1	0.256733	0.743267
2	0.112146	0.887854
3	0.168994	0.831006
4	0.026376	0.973624

Fig-1.32

Confusion Matrix:-

```
[242,  90]
[ 82, 653]
```

Fig-1.33

Accuracy & Classification: -

```
NB1_train_acc=NB_model.score(X_train,Y_train)
print(NB1_train_acc)
```

```
0.8388003748828491
```

```
print(classification_report(Y_train,ytrain_predict_NB1))
```

```
              precision    recall  f1-score   support

           0       0.75      0.73      0.74       332
           1       0.88      0.89      0.88       735

    accuracy                           0.84      1067
   macro avg       0.81      0.81      0.81      1067
weighted avg       0.84      0.84      0.84      1067
```

Fig-1.34

Naïve Bayes Test:-

The Predicted Classes and Probs - NB_Model:-

	0	1
0	0.990544	0.009456
1	0.874555	0.125445
2	0.402754	0.597246
3	0.566358	0.433642
4	0.231714	0.768286

Fig-1.35

Confusion Matrix:-

```
[ 94,  36],
[ 46, 282]]
```

Fig-1.36

Accuracy & Classification: -

```
NB1_test_acc=NB_model.score(X_test,Y_test)
print(NB1_test_acc)
```

```
0.8209606986899564
```

```
print(classification_report(Y_test,ytest_predict4_NB1))
```

```
              precision    recall  f1-score   support

           0       0.67      0.72      0.70       130
           1       0.89      0.86      0.87       328

    accuracy                           0.82       458
   macro avg       0.78      0.79      0.78       458
weighted avg       0.83      0.82      0.82       458
```

Fig-1.37

Conclusion | insight:

1. The model scores are good for both the models.
2. Accuracy, Recall, Precision and F1-Score, all the parameters are having good values for both the models.

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

Model Tuning-

GridSearchCV for Logistic regression:-

```
grid_search1.best_params_
```

```
{'max_iter': 500,
 'n_jobs': 3,
 'penalty': 'none',
 'solver': 'lbfgs',
 'tol': 0.1}
```

```
best_model_LR
```

```
LogisticRegression(max_iter=500, n_jobs=3, penalty='none', tol=0.1)
```

Fig-1.6.1

Train & Test :-

```
### Getting the Predicted Classes and Probs
ytrain_predict1_prob=best_model_LR.predict_proba(X_train)
pd.DataFrame(ytrain_predict1_prob).head()
```

	0	1
0	0.630689	0.369311
1	0.180593	0.819407
2	0.191808	0.808192
3	0.168730	0.831270
4	0.053652	0.946348

```
### Getting the Predicted Classes and Probs
ytest_predict1_prob=best_model_LR.predict_proba(X_test)
pd.DataFrame(ytest_predict1_prob).head()
```

	0	1
0	0.930306	0.069694
1	0.696465	0.303535
2	0.322962	0.677038
3	0.473096	0.526904
4	0.151897	0.848103

Fig-1.6.2

GridSearchCV for LDA model for Model Tuning

```
grid_search2.best_params_
```

```
{'n_components': 0, 'solver': 'svd', 'tol': 0.0001}
```

Fig-1.6.3

Train & Test :-

```
best_modal_LDA=grid_search2.best_estimator_
best_modal_LDA
```

```
LinearDiscriminantAnalysis(n_components=0)
```

```
ytrain_predict2_prob=best_modal_LDA.predict_proba(X_train)
pd.DataFrame(ytrain_predict2_prob).head()
```

	0	1
0	0.655725	0.344275
1	0.156789	0.843211
2	0.186375	0.813625
3	0.136064	0.863936
4	0.040668	0.959332

```
ytest_predict2_prob=best_modal_LDA.predict_proba(X_test)
pd.DataFrame(ytest_predict2_prob).head()
```

	0	1
0	0.950899	0.049101
1	0.745815	0.254185
2	0.321758	0.678242
3	0.484760	0.515240
4	0.135276	0.864724

Fig-1.6.4

GridSearchCV for Naïve Bayes Model model for Model tuning

```
grid_search3.best_params_
```

```
{'var_smoothing': 1e-06}
```

```
best_modal_NB=grid_search3.best_estimator_
best_modal_NB
```

```
GaussianNB(var_smoothing=1e-06)
```

Fig-1.6.5

Train & Test :-

```
ytrain_predict3_prob=best_modal_NB.predict_proba(X_train)
pd.DataFrame(ytrain_predict3_prob).head()
```

	0	1
0	0.643755	0.356245
1	0.276513	0.723487
2	0.163415	0.836585
3	0.168703	0.831297
4	0.042195	0.957805

```
ytest_predict3_prob=best_modal_NB.predict_proba(X_test)
pd.DataFrame(ytest_predict3_prob).head()
```

	0	1
0	0.974577	0.025423
1	0.834625	0.165375
2	0.340345	0.659655
3	0.526856	0.473144
4	0.275650	0.724350

Fig-1.6.6

GridSearchCV for KNN model for Model tuning

```
grid_search4.best_params_
```

```
{'leaf_size': 1, 'n_neighbors': 7, 'p': 1}
```

```
best_model_KNN = grid_search4.best_estimator_
best_model_KNN
```

```
KNeighborsClassifier(leaf_size=1, n_neighbors=7, p=1)
```

Fig-1.6.7

Train & Test :-

```
### Getting the Predicted Classes and Probs
ytrain_predict4_prob=best_model_KNN.predict_proba(x_train)
pd.DataFrame(ytrain_predict4_prob).head()
```

	0	1
0	0.571429	0.428571
1	0.142857	0.857143
2	0.571429	0.428571
3	0.142857	0.857143
4	0.285714	0.714286

```
### Getting the Predicted Classes and Probs
ytest_predict4_prob=best_model_KNN.predict_proba(x_test)
pd.DataFrame(ytest_predict4_prob).head()
```

	0	1
0	0.428571	0.571429
1	0.428571	0.571429
2	0.571429	0.428571
3	0.571429	0.428571
4	0.142857	0.857143

Fig-1.6.8

GridSearchCV for Bagging for Model tuning

```
grid_search5.best_params_
```

```
{'base_estimator__max_depth': 4, 'max_samples': 0.5}
```

```
best_model_BAG = grid_search5.best_estimator_
best_model_BAG
```

```
BaggingClassifier(base_estimator=DecisionTreeClassifier(max_depth=4),
                  max_features=0.5, max_samples=0.5, n_estimators=100)
```

Fig-1.6.9

Train & Test :-

```
### Getting the Predicted Classes and Probs
ytrain_predict5_prob=best_model_BAG.predict_proba(X_train)
pd.DataFrame(ytrain_predict5_prob).head()
```

	0	1
0	0.408592	0.591408
1	0.308009	0.691991
2	0.279326	0.720674
3	0.187223	0.812777
4	0.303350	0.696650

```
ytest_predict5_prob=best_model_BAG.predict_proba(X_test)
pd.DataFrame(ytest_predict5_prob).head()
```

	0	1
0	0.593969	0.406031
1	0.564113	0.435887
2	0.321651	0.678349
3	0.454448	0.545552
4	0.283384	0.716616

Fig-1.6.10

GridSearchCV for Boosting ADA for Model tuning

```
grid_search6.best_params_
```

```
{'learning_rate': 0.1, 'n_estimators': 230}
```

```
best_model_BOS = grid_search6.best_estimator_
best_model_BOS
```

```
AdaBoostClassifier(learning_rate=0.1, n_estimators=230)
```

Fig-1.6.11

Train & Test :-

```
### Getting the Predicted Classes and Probs
ytrain_predict6_prob=best_model_BOS.predict_proba(X_train)
pd.DataFrame(ytrain_predict6_prob).head()
```

	0	1
0	0.503558	0.496442
1	0.491953	0.508047
2	0.487713	0.512287
3	0.481439	0.518561
4	0.489638	0.510362

```
ytest_predict6_prob=best_model_BOS.predict_proba(X_test)
pd.DataFrame(ytest_predict6_prob).head()
```

	0	1
0	0.516017	0.483983
1	0.511696	0.488304
2	0.494338	0.505662
3	0.502055	0.497945
4	0.489286	0.510714

Fig-1.6.12

GridSearchCV for Gradient Boosting for Model Tuning

```
grid_search7.best_params_
```

```
{'learning_rate': 0.005, 'n_estimators': 500}
```

```
best_model_GBM = grid_search7.best_estimator_
best_model_GBM
```

```
GradientBoostingClassifier(learning_rate=0.005, max_depth=4,
                           max_features='sqrt', n_estimators=500,
                           random_state=10, subsample=1)
```

Fig-1.6.13

Train & Test :-

```
### Getting the Predicted Classes and Probs
ytrain_predict7_prob=best_model_GBM.predict_proba(X_train)
pd.DataFrame(ytrain_predict7_prob).head()
```

	0	1
0	0.481270	0.518730
1	0.298229	0.701771
2	0.250246	0.749754
3	0.150825	0.849175
4	0.283924	0.716076

```
### Getting the Predicted Classes and Probs
ytest_predict7_prob=best_model_GBM.predict_proba(X_test)
pd.DataFrame(ytest_predict7_prob).head()
```

	0	1
0	0.716553	0.283447
1	0.788920	0.211080
2	0.230296	0.769704
3	0.350325	0.649675
4	0.244124	0.755876

Fig-1.6.14

Conclusion | insight: GridSearchCV for using all seven modal for camper which modal performance best of case of election voting Result. Available data are all modal are mention in GridsearchCV based .

Q1.7 Performance Metrics: Check the performance of Predictions on Train and Test sets using Accuracy, Confusion Matrix, Plot ROC curve and get ROC_AUC score for each model. Final Model: Compare the models and write inference which model is best/optimized.

Logistic Regression- Confusion Matrix Train Accuracy:-

```
[229, 103]
[ 70, 665]
```

	precision	recall	f1-score	support
0	0.77	0.69	0.73	332
1	0.87	0.90	0.88	735
accuracy			0.84	1067
macro avg	0.82	0.80	0.81	1067
weighted avg	0.83	0.84	0.84	1067

Fig-1.7.0

ROC Train:-

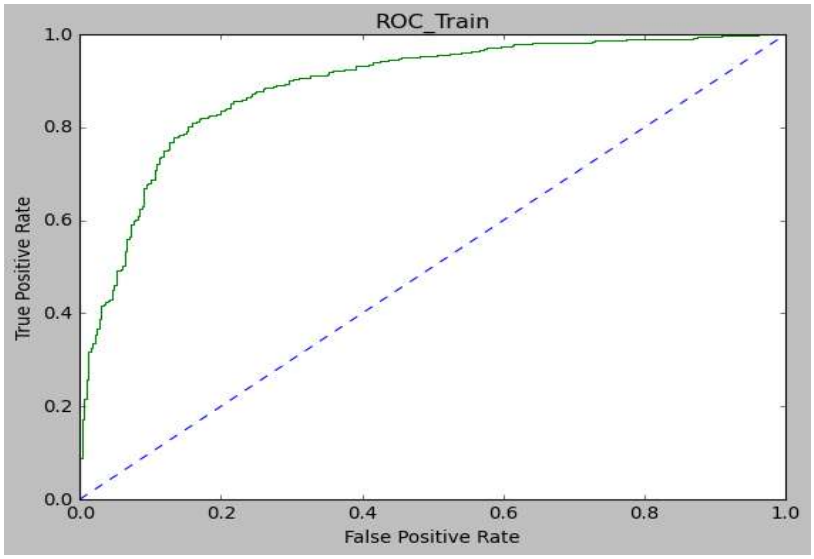

Fig-1.7.1

Logistic Regression- Confusion Matrix Test Accuracy

```
[ 85,  45]
[ 36, 292]
```

	precision	recall	f1-score	support
0	0.70	0.65	0.68	130
1	0.87	0.89	0.88	328
accuracy			0.82	458
macro avg	0.78	0.77	0.78	458
weighted avg	0.82	0.82	0.82	458

Fig-1.7.2

ROC Test:-

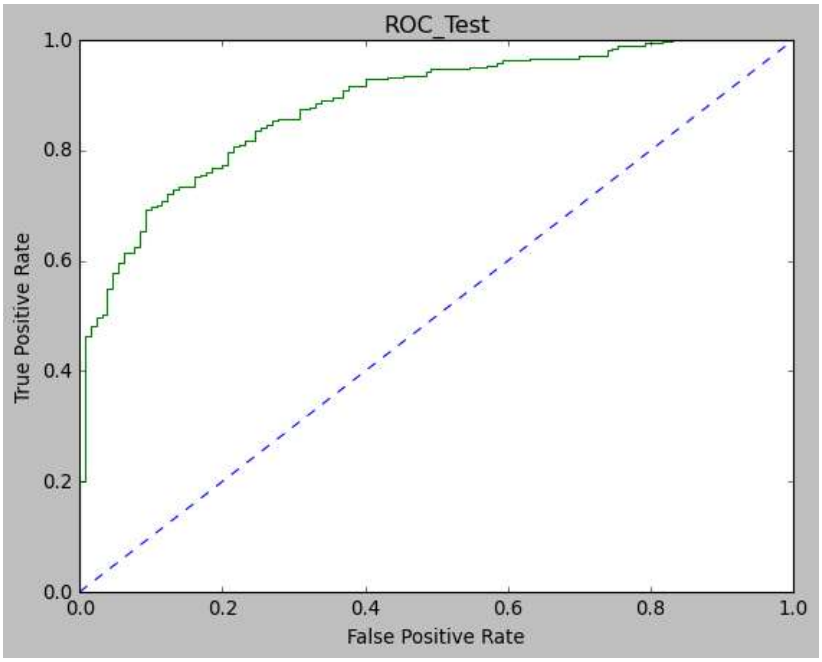

Fig-1.7.3

LDA- Confusion Matrix Train- Accuracy

```
[235,  97]
[ 75, 660]
```

	precision	recall	f1-score	support
0	0.76	0.71	0.73	332
1	0.87	0.90	0.88	735
accuracy			0.84	1067
macro avg	0.81	0.80	0.81	1067
weighted avg	0.84	0.84	0.84	1067

Fig-1.7.4

ROC Train:-

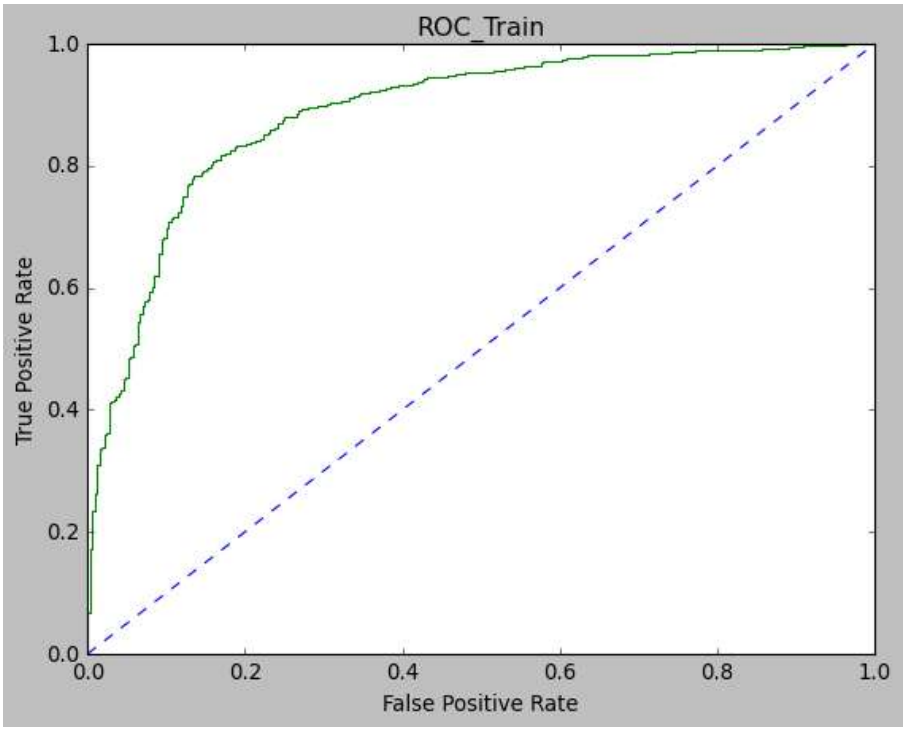

Fig-1.7.5

LDA- Confusion Matrix Test Accuracy

```
[ 86,  44]
[ 39, 289]
```

	precision	recall	f1-score	support
0	0.69	0.66	0.67	130
1	0.87	0.88	0.87	328
accuracy			0.82	458
macro avg	0.78	0.77	0.77	458
weighted avg	0.82	0.82	0.82	458

Fig-1.7.6

ROC Test:-

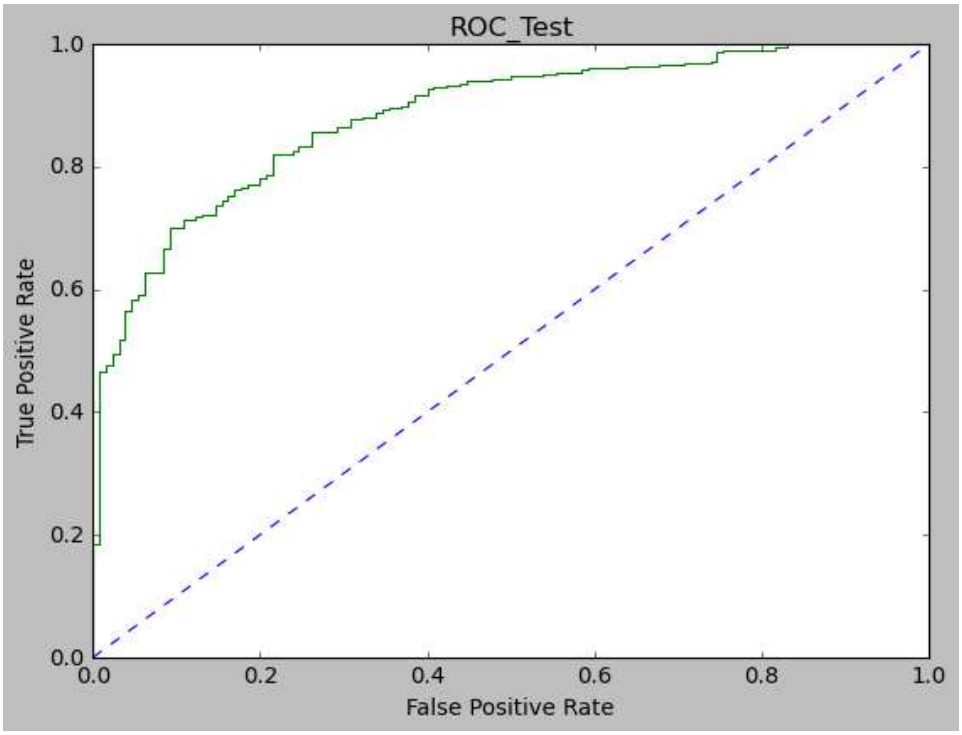

Fig-1.7.7

Naïve Bayes Model-Confusion Matrix Train Accuracy

```
[237,  95]
[ 79, 656]
```

	precision	recall	f1-score	support
0	0.75	0.71	0.73	332
1	0.87	0.89	0.88	735
accuracy			0.84	1067
macro avg	0.81	0.80	0.81	1067
weighted avg	0.84	0.84	0.84	1067

Fig-1.7.8

ROC Train:-

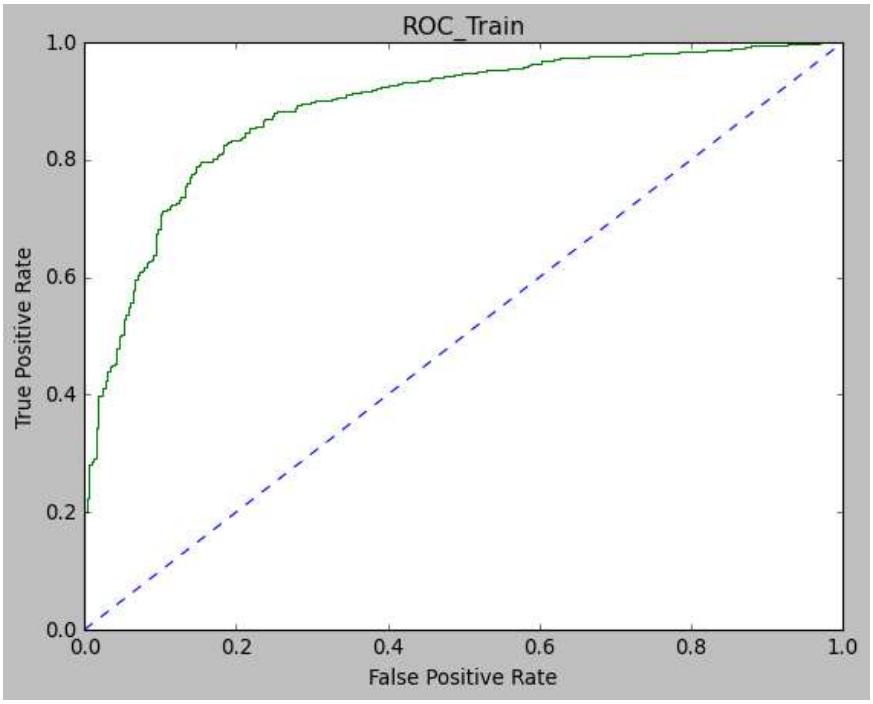

Fig-1.7.9

Naïve Bayes Model-Confusion Matrix Test Accuracy

```
[ 94,  36]
[ 41, 287]
```

	precision	recall	f1-score	support
0	0.70	0.72	0.71	130
1	0.89	0.88	0.88	328
accuracy			0.83	458
macro avg	0.79	0.80	0.80	458
weighted avg	0.83	0.83	0.83	458

Fig-1.7.10

ROC Test:-

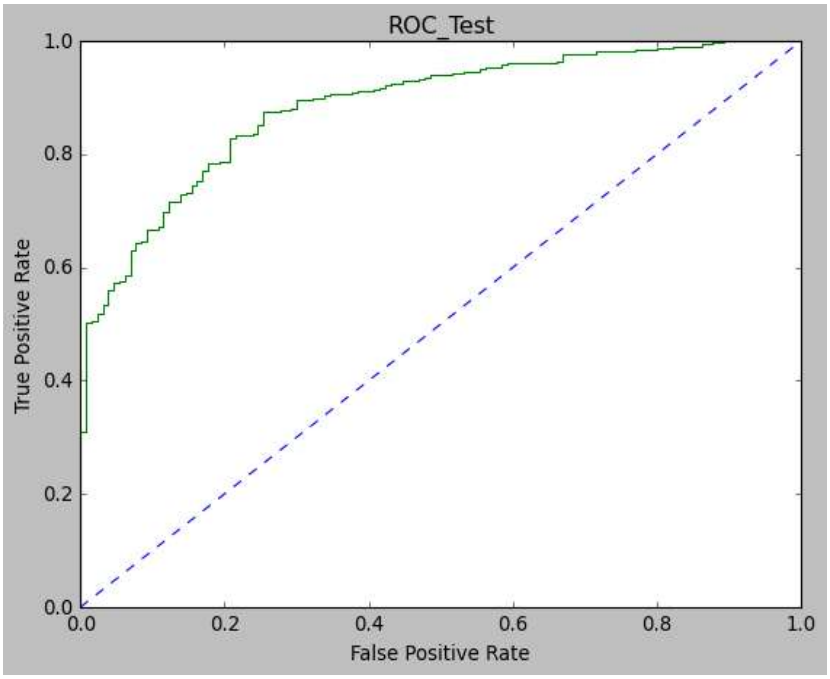

Fig-1.7.11

KNN-Confusion Matrix Train Accuracy

```
[173, 178]
[ 55, 737]
```

	precision	recall	f1-score	support
0	0.76	0.49	0.60	351
1	0.81	0.93	0.86	792
accuracy			0.80	1143
macro avg	0.78	0.71	0.73	1143
weighted avg	0.79	0.80	0.78	1143

Fig-1.7.12

ROC Train:-

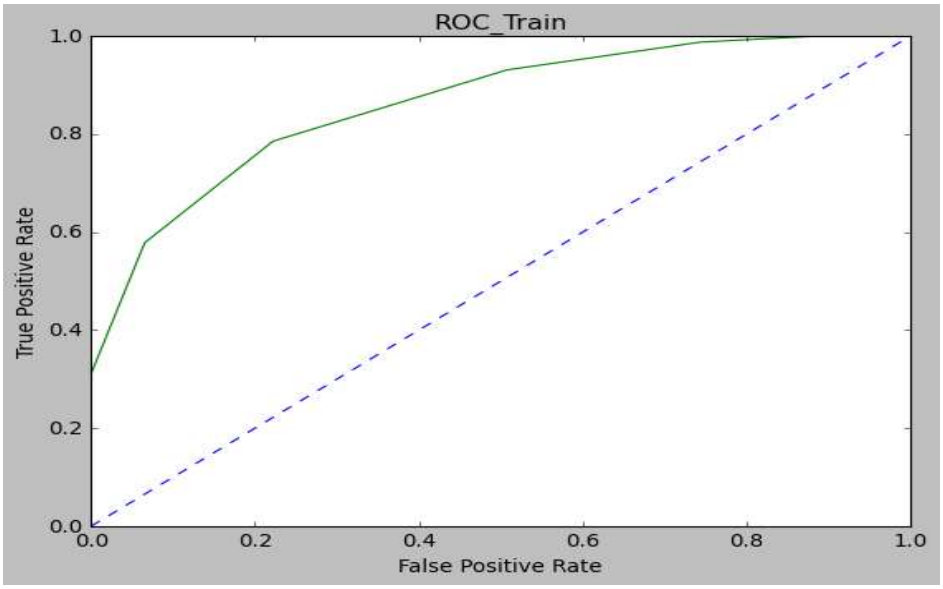

Fig-1.7.13

KNN-Confusion Matrix Test Accuracy

```
[ 38,  73]
[ 35, 236]
```

	precision	recall	f1-score	support
0	0.52	0.34	0.41	111
1	0.76	0.87	0.81	271
accuracy			0.72	382
macro avg	0.64	0.61	0.61	382
weighted avg	0.69	0.72	0.70	382

Fig-1.7.14

ROC Test:-

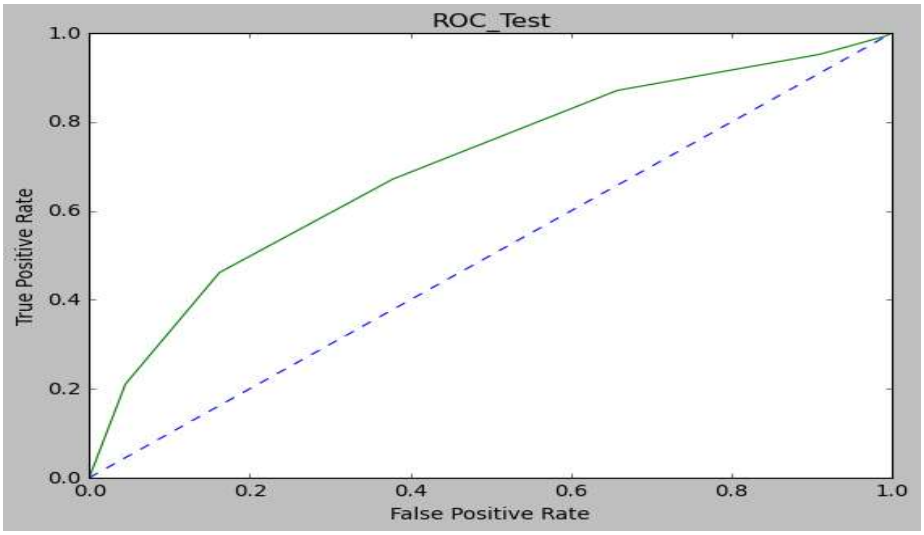

Fig-1.7.15

Bagging Train-Confusion Matrix Train Accuracy

```
[194, 138]
[ 29, 706]
```

	precision	recall	f1-score	support
0	0.87	0.58	0.70	332
1	0.84	0.96	0.89	735
accuracy			0.84	1067
macro avg	0.85	0.77	0.80	1067
weighted avg	0.85	0.84	0.83	1067

Fig-1.7.16

ROC Train:-

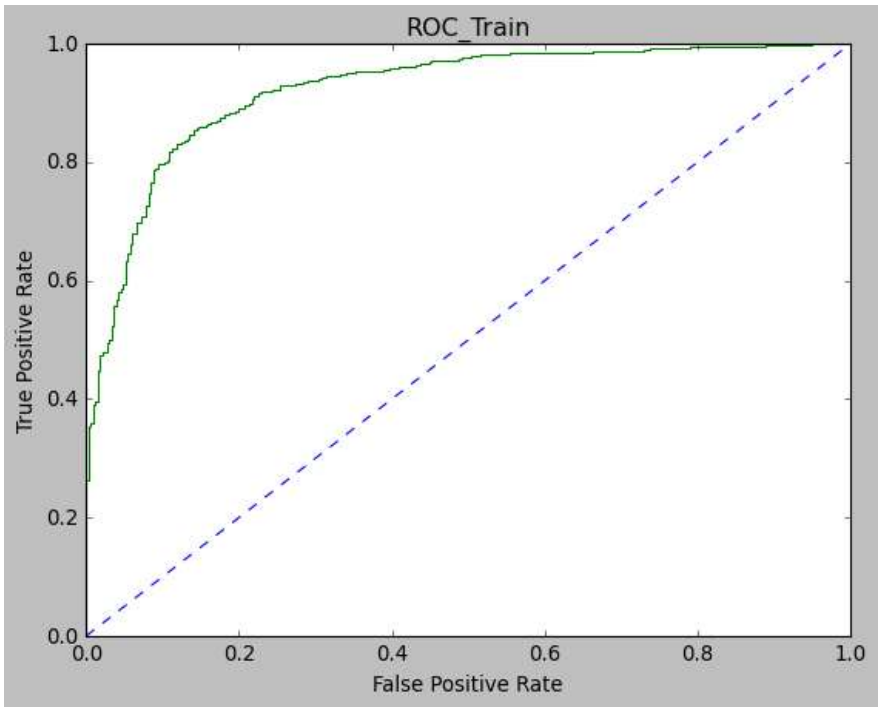

Fig-1.7.18

Bagging Train-Confusion Matrix Test Accuracy

```
[ 71,  59]
[ 21, 307]
```

	precision	recall	f1-score	support
0	0.77	0.55	0.64	130
1	0.84	0.94	0.88	328
accuracy			0.83	458
macro avg	0.81	0.74	0.76	458
weighted avg	0.82	0.83	0.82	458

Fig-1.7.19

ROC Test:-

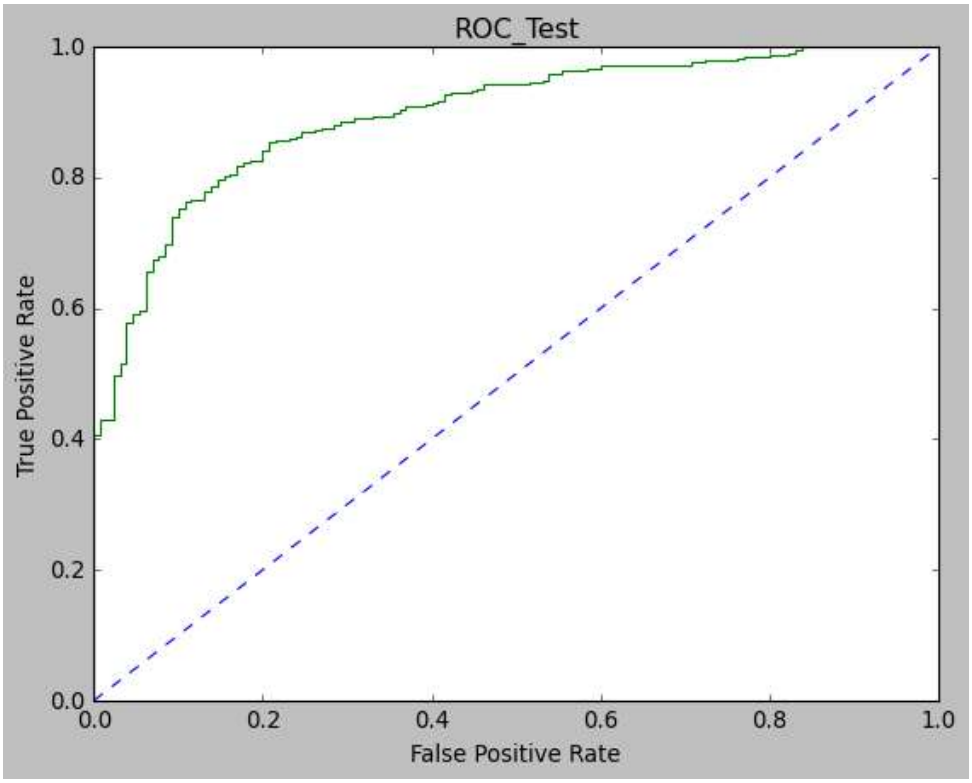

Fig-1.7.20

Boosting ADA Confusion Matrix Train Accuracy

```
[232, 100]
[ 60, 675]
```

	precision	recall	f1-score	support
0	0.79	0.70	0.74	332
1	0.87	0.92	0.89	735
accuracy			0.85	1067
macro avg	0.83	0.81	0.82	1067
weighted avg	0.85	0.85	0.85	1067

Fig-1.7.21

ROC Train:-

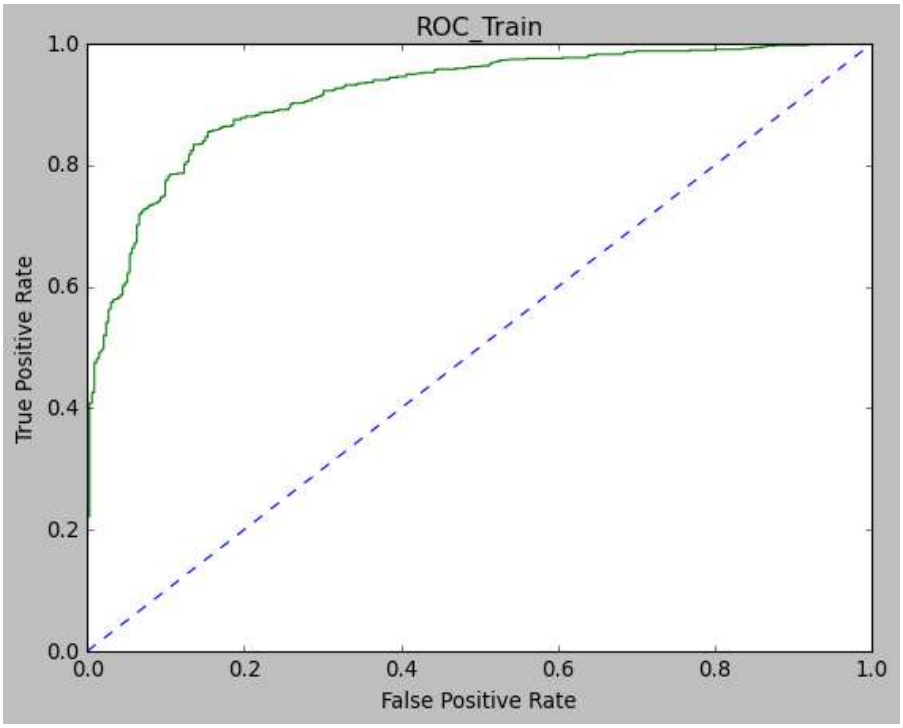

Fig-1.7.22

Boosting ADA Confusion Matrix Test Accuracy

```
[ 84,  46]
[ 35, 293]
```

	precision	recall	f1-score	support
0	0.71	0.65	0.67	130
1	0.86	0.89	0.88	328
accuracy			0.82	458
macro avg	0.79	0.77	0.78	458
weighted avg	0.82	0.82	0.82	458

Fig-1.7.23

ROC Test:-

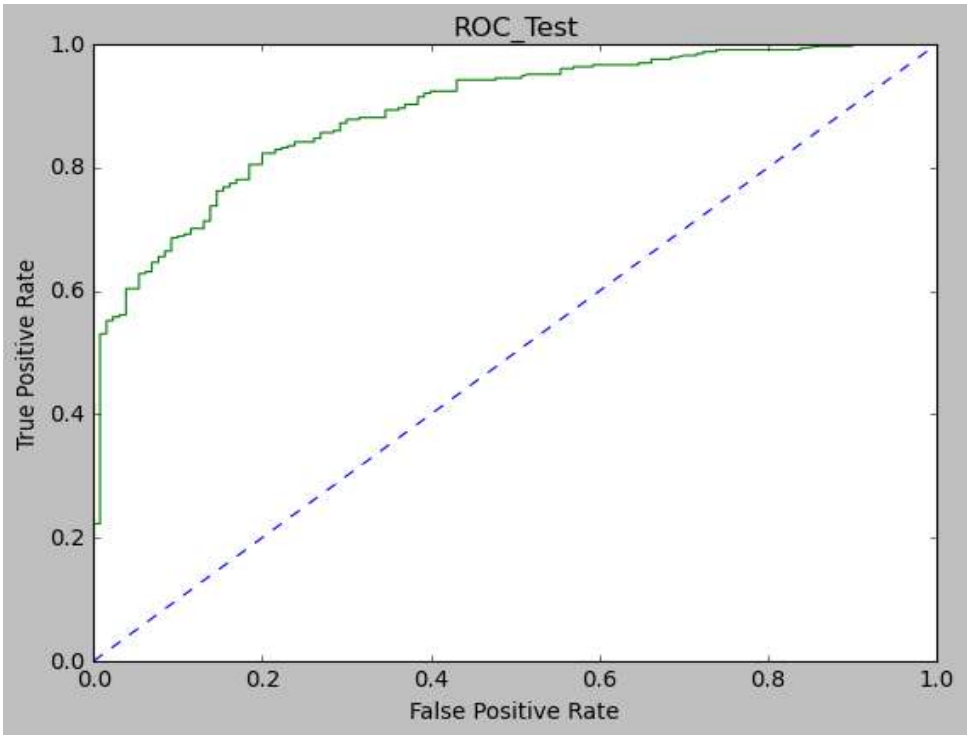

Fig-1.7.24

Gradient Boosting Confusion Matrix Train Accuracy

```
[253,  79],
[ 44, 691]]
```

	precision	recall	f1-score	support
0	0.85	0.76	0.80	332
1	0.90	0.94	0.92	735
accuracy			0.88	1067
macro avg	0.87	0.85	0.86	1067
weighted avg	0.88	0.88	0.88	1067

Fig-1.7.25

ROC Train:-

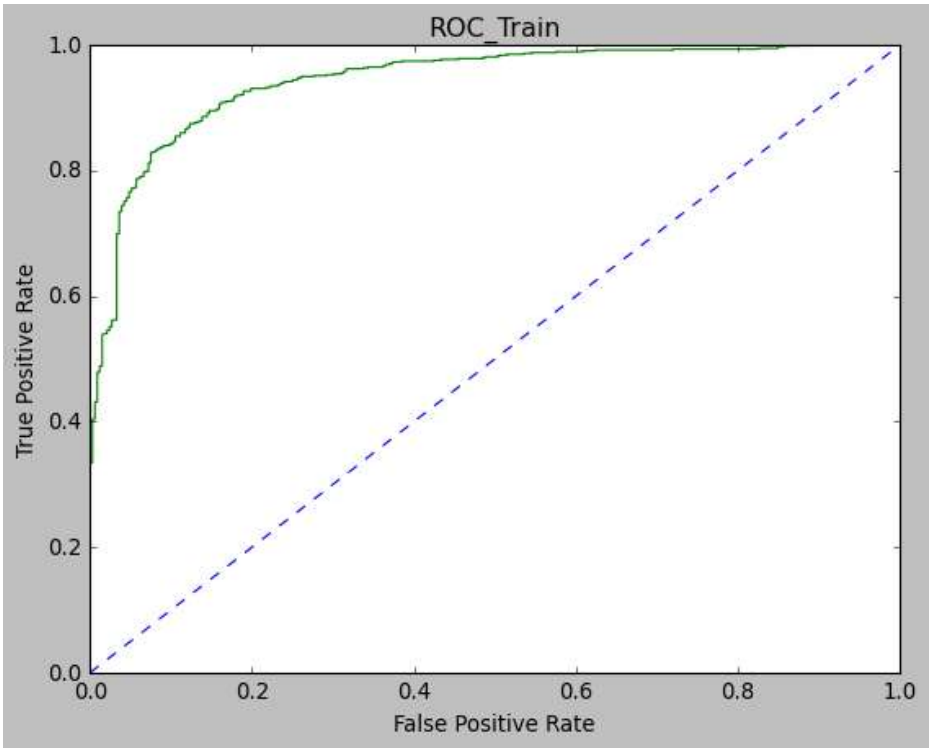

Fig-1.7.26

Gradient Boosting Confusion Matrix Test Accuracy

```
[ 91,  39]
[ 37, 291]
```

	precision	recall	f1-score	support
0	0.71	0.70	0.71	130
1	0.88	0.89	0.88	328
accuracy			0.83	458
macro avg	0.80	0.79	0.79	458
weighted avg	0.83	0.83	0.83	458

Fig-1.7.27

ROC Train:-

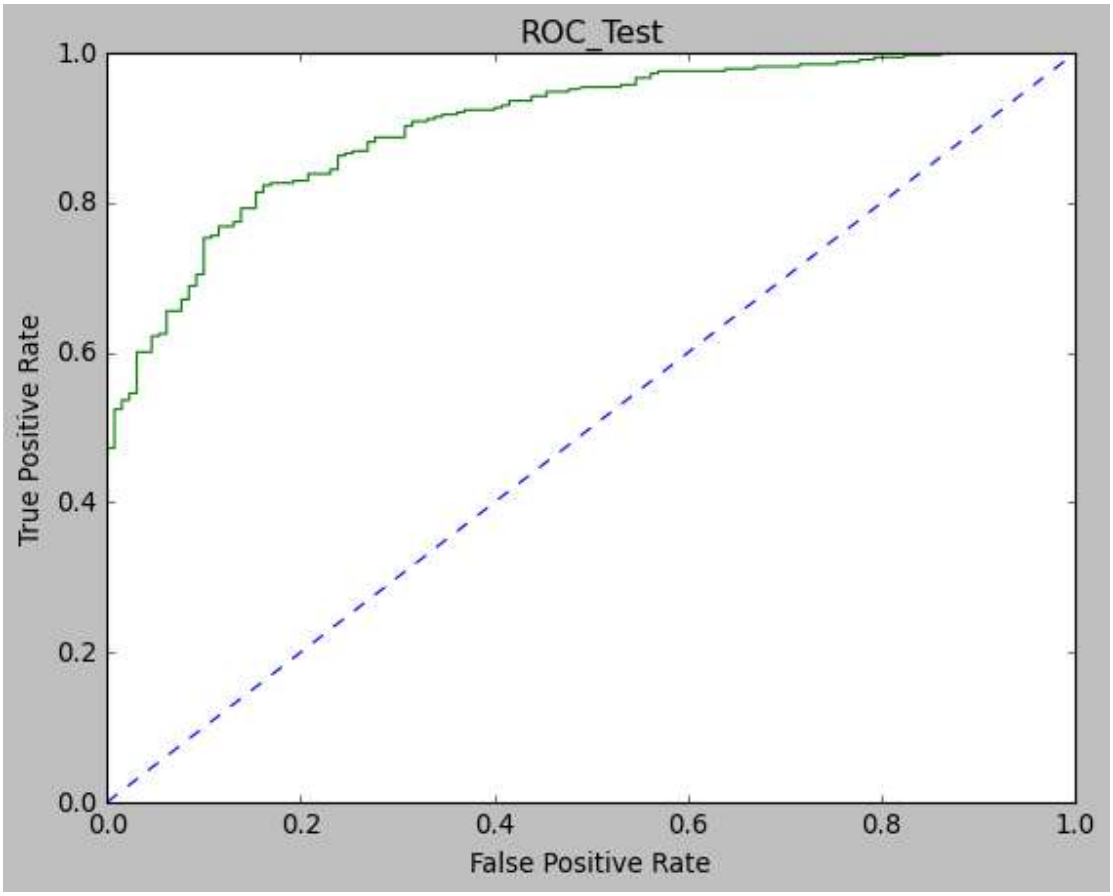

Fig-1.7.28

Final Model: - Compare the models and write inference which model is best/optimized.

ALL Modal:-

	LR Train	LR Test	LDA Train	LDA Test	Naïve Bayes-NB Train	Naïve Bayes-NB Test	KNN Train	KNN Test	Bagging-BAG Train	Bagging-BAG Test	Boosting ADA-BOS Train	Boosting ADA-BOS Test	Gradient Boosting-GBM Train	Gradient Boosting-GBM Test
Accuracy	0.84	0.82	0.84	0.82	0.84	0.83	0.80	0.72	0.84	0.83	0.85	0.82	0.88	0.83
AUC	0.89	0.88	0.89	0.88	0.89	0.89	0.86	0.70	0.92	0.90	0.92	0.89	0.94	0.90
Recall	0.90	0.89	0.90	0.88	0.89	0.88	0.93	0.87	0.96	0.94	0.92	0.89	0.94	0.89
Precision	0.87	0.87	0.87	0.87	0.87	0.89	0.81	0.76	0.84	0.84	0.87	0.86	0.90	0.88
F1 Score	0.88	0.88	0.88	0.87	0.88	0.88	0.86	0.81	0.89	0.88	0.89	0.88	0.92	0.88

Fig-1.7.28.a

Conclusion | insight: All modal accuracy, AUC, Recall, Precision and F1 Score are same time compare in this format rest result shown in fig-1.7.28.a.

ROC Curve for the 7 models on the Train data

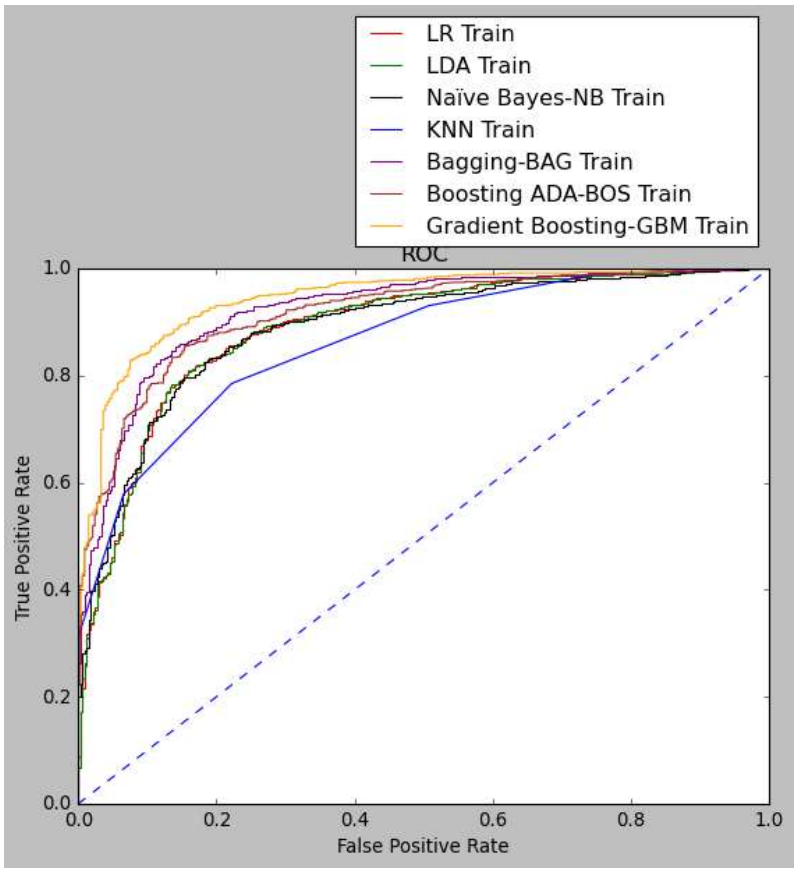

Fig-1.7.29

ROC Curve for the 7 models on the Test data

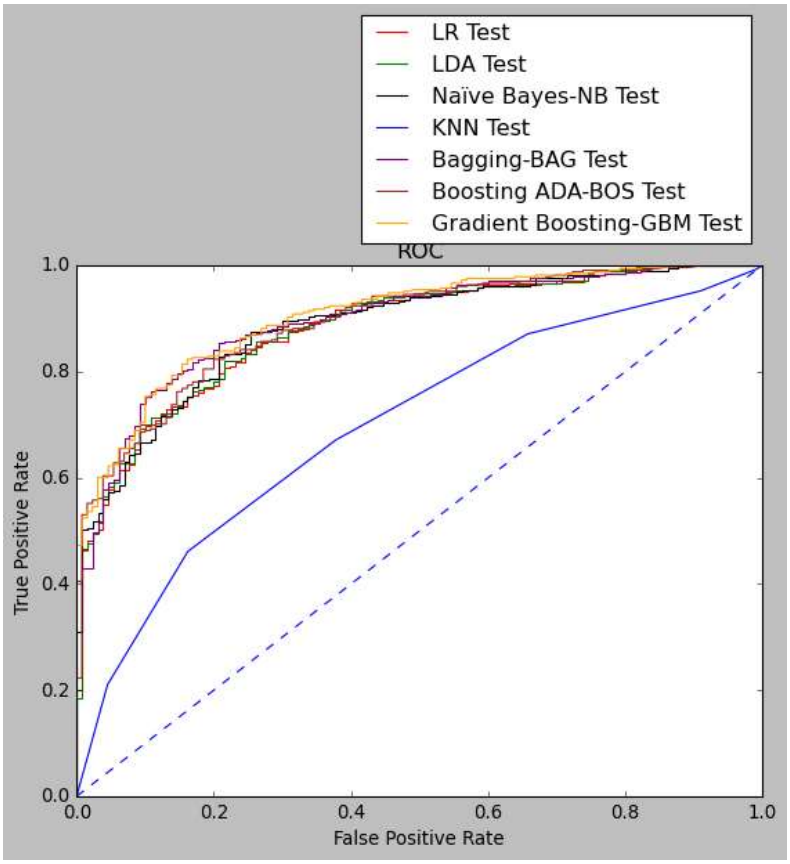

Fig-1.7.30

Conclusion | insight:

1. Gradient Boosting model performs the best with 89% train accuracy. And also have 90% precision and 96% recall which is better than any other models that we have performed in here with the Election dataset.

2. Rest all the models are more or less have same accuracy of 88%

3. All the models have a decent score except bagging classifier, without tuning model was overfit with 100% scores in train set.

4. ROC is also show in fig performance of all modal in one pic all behaviours of Modals.

Q 1.8 Based on these predictions, what are the insights?

Conclusion | insight:

1. Comparing all the models we see that KNN model and Gradient Boosting model provides best results.

2. We can say Gradient boosting model outperforms all the models and giving best results to be specific.
3. We observe Labour has highest possibility of winning

4. Labour has higher voting possibility among all age groups except for very old people.

5. Irrespective of the political knowledge levels or gender Labour has an edge on higher votes.

6. The important variable in predicting the dependent variables are ‘Hague’ and ‘Blair’

Project-2

United States of America speeches from inaugural corpora.

Table of Contents

Contents

Table & Figures

Executive Summary:-

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

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

Introduction:-

The purpose of this whole exercise is to explore the dataset. Do the exploratory data analysis. Explore the dataset of file text and pic and any of data analysis of using method of NLTK all speech was available in python we are using and show what type of methored are using build best solution.

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

```
'Mr. Vice President, Mr. Speaker, Mr. Chief Justice, Senator Cook, Mrs. Eisenhower, and my fellow citizens of this great and go
od country we share together:\n\nWhen we met here four years ago, America was bleak in spirit, depressed by the prospect of see
mingly 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 a
bout to enter will not be what other postwar periods have so often been: a time of retreat and isolation that leads to stagnati
on at home and invites new danger abroad.\n\nLet us resolve that this will be what it can become: a time of great responsibilit
ies greatly borne, in which we renew the spirit and the promise of America as we enter our third century as a nation.\n\nThis p
ast year saw far-reaching results from our new policies for peace. By continuing to revitalize our traditional friendships, and
by our missions to Peking and to Moscow, we were able to establish the base for a new and more durable pattern of relationships
among the nations of the world. Because of America\'s bold initiatives, 1972 will be long remembered as the year of the greates
t progress since the end of World War II toward a lasting peace in the world.\n\nThe peace we seek in the world is not the flim
sy peace which is merely an interlude between wars, but a peace which can endure for generations to come.\n\nIt is important th
at we understand both the necessity and the limitations of America\'s role in maintaining that peace.\n\nUnless we in America w
ork to preserve the peace, there will be no peace.\n\nUnless we in America work to preserve freedom, there will be no freedo
m.\n\nBut let us clearly understand the new nature of America\'s role, as a result of the new policies we have adopted over the
se past four years.\n\nWe shall respect our treaty commitments.\n\nWe shall support vigorously the principle that no country ha
s the right to impose its will or rule on another by force.\n\nWe shall continue, in this era of negotiation, to work for the l
imitation of nuclear arms, and to reduce the danger of confrontation between the great powers.\n\nWe shall do our share in defe
nding peace and freedom in the world. But we shall expect others to do their share.\n\nThe time has passed when America will ma
ke every other nation\'s conflict our own, or make every other nation\'s future our responsibility, or presume to tell the peop
le of other nations how to manage their own affairs.\n\nJust as we respect the right of each nation to determine its own futur
e, we also recognize the responsibility of each nation to secure its own future.\n\nJust as America\'s role is indispensable in
preserving the world\'s peace, so is each nation\'s role indispensable in preserving its own peace.\n\nTogether with the rest o
f the world, let us resolve to move forward from the beginnings we have made. Let us continue to bring down the walls of hostil
ity which have divided the world for too long, and to build in their place bridges of understanding -- so that despite profound
differences between systems of government, the people of the world can be friends.\n\nLet us build a structure of peace in the
```

Fig-2.1

Conclusion | insight: Three speech is shown, after importing the text file, we would first count the total number of characters in each file separately. Below is the code to count the char from each file. With the output along with the screenshot.

Number of Characters in each file:-

```
Number of characters in Roosevelt file 7571
Number of characters in Kennedy file 7618
Number of characters in Nixon file 9991
```

Fig-2.2

Conclusion | insight: All are shown in fig -2.2.

Number of words in each text file:-

Conclusion | insight: Below we are counting the total number of words from each file separately. Here we are using the split() to split up the words based on space between each word and we are counting the total number of words by using the len() function.

```
Number of word in Roosevelt file : 1360

Number of word in kennedy file : 1390

Number of word in Nixon file : 1819
```

Fig-2.3

Number of Sentences:-

Conclusion | insight: Below we are counting the total number of sentence in each text file, by using lambda function. We are using pd.Dataframe to move the data as dictionary and then with lambda function we are checking each sentece which ends with "." Using endswith() function and the below code and output is as below.

a) Roosevelt

	Text	sentences
0	On each national day of inauguration since 178...	67

b) Kennedy

	Text	sentences
0	Vice President Johnson, Mr. Speaker, Mr. Chief...	52

c) Nixon

	Text	sentences
0	Mr. Vice President, Mr. Speaker, Mr. Chief Jus...	68

Fig-2.4

Conclusion | insight: Most of the sentences are available in Nixon speech as per fig 2.4 was shown.

Q 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.

Conclusion | insight: All are helping verb are shown in below mention in Snp using command in python note pad all three speech using all words shown some small Snp mention in blow.

a) Roosevelt

```
['On', 'each', 'national', 'day', 'of', 'inauguration', 'since', '1789', ',', 'the', 'people', 'have', 'renewed', 'their', 's
ense', 'of', 'dedication', 'to', 'the', 'United', 'States', '.', 'In', 'Washington', "'s", 'day', 'the', 'task', 'of', 'the',
'people', 'was', 'to', 'create', 'and', 'weld', 'together', 'a', 'nation', '.', 'In', 'Lincoln', "'s", 'day', 'the', 'task',
'of', 'the', 'people', 'was', 'to', 'preserve', 'that', 'Nation', 'from', 'disruption', 'from', 'within', '.', 'In', 'this',
'day', 'the', 'task', 'of', 'the', 'people', 'is', 'to', 'save', 'that', 'Nation', 'and', 'its', 'institutions', 'from', 'dis
ruption', 'from', 'without', '.', 'To', 'us', 'there', 'has', 'come', 'a', 'time', ',', 'in', 'the', 'midst', 'of', 'swift',
'happenings', ',', 'to', 'pause', 'for', 'a', 'moment', 'and', 'take', 'stock', '--', 'to', 'recall', 'what', 'our', 'place',
'in', 'history', 'has', 'been', ',', 'and', 'to', 'rediscover', 'what', 'we', 'are', 'and', 'what', 'we', 'may', 'be', '.',
'If', 'we', 'do', 'not', ',', 'we', 'risk', 'the', 'real', 'peril', 'of', 'inaction', '.', 'Lives', 'of', 'nations', 'are',
'determined', 'not', 'by', 'the', 'count', 'of', 'years', ',', 'but', 'by', 'the', 'lifetime', 'of', 'the', 'human', 'spiri
t', '.', 'The', 'life', 'of', 'a', 'man', 'is', 'three-score', 'years', 'and', 'ten', ':', 'a', 'little', 'more', ',', 'a',
'little', 'less', '.', 'The', 'life', 'of', 'a', 'nation', 'is', 'the', 'fullness', 'of', 'the', 'measure', 'of', 'its', 'wil
l', 'to', 'live', '.', 'There', 'are', 'men', 'who', 'doubt', 'this', '.', 'There', 'are', 'men', 'who', 'believe', 'that',
'democracy', ',', 'as', 'a', 'form', 'of', 'Government', 'and', 'a', 'frame', 'of', 'life', ',', 'is', 'limited', 'or', 'meas
ured', 'by', 'a', 'kind', 'of', 'mystical', 'and', 'artificial', 'fate', 'that', ',', 'for', 'some', 'unexplained', 'reason',
',', 'tyranny', 'and', 'slavery', 'have', 'become', 'the', 'surging', 'wave', 'of', 'the', 'future', '--', 'and', 'that', 'fr
eedom', 'is', 'an', 'ebbing', 'tide', '.', 'But', 'we', 'Americans', 'know', 'that', 'this', 'is', 'not', 'true', '.', 'Eigh
t', 'years', 'ago', ',', 'when', 'the', 'life', 'of', 'this', 'Republic', 'seemed', 'frozen', 'by', 'a', 'fatalistic', 'terro
r', ',', 'we', 'proved', 'that', 'this', 'is', 'not', 'true', '.', 'We', 'were', 'in', 'the', 'midst', 'of', 'shock', '--',
'but', 'we', 'acted', '.', 'We', 'acted', 'quickly', '.', 'boldly', '.', 'decisively', '.', 'These', 'later', 'years', 'hav
```

Fig-2.2.1

b) Kennedy

```
['Vice', 'President', 'Johnson', ',', 'Mr.', 'Speaker', ',', 'Mr.', 'Chief', 'Justice', ',', 'President', 'Eisenhower', ',',
'Vice', 'President', 'Nixon', ',', 'President', 'Truman', ',', 'reverend', 'clergy', ',', 'fellow', 'citizens', ',', 'we', 'o
bserve', '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', 'fo
rebears', 'l', 'prescribed', 'nearly', 'a', 'century', 'and', 'three', 'quarters', 'ago', '.', 'The', 'world', 'is', 'very',
'different', 'now', '.', 'For', 'man', 'holds', 'in', 'his', 'mortal', 'hands', 'the', 'power', 'to', 'abolish', 'all', 'form
s', 'of', 'human', 'poverty', 'and', 'all', 'forms', 'of', 'human', 'life', '.', 'And', 'yet', 'the', 'same', 'revolutionar
y', 'beliefs', 'for', 'which', 'our', 'forebears', 'fought', 'are', 'still', 'at', 'issue', 'around', 'the', 'globe', '--',
'the', 'belief', 'that', 'the', 'rights', 'of', 'man', 'come', 'not', 'from', 'the', 'generosity', 'of', 'the', 'state', ',',
'but', 'from', 'the', 'hand', 'of', 'God', '.', 'We', 'dare', 'not', 'forget', 'today', 'that', 'we', 'are', 'the', 'heirs',
'of', 'that', 'first', 'revolution', '.', 'Let', 'the', 'word', 'go', 'forth', 'from', 'this', 'time', 'and', 'place', ',',
'to', 'friend', 'and', 'foe', 'alike', ',', 'that', 'the', 'torch', 'has', 'been', 'passed', 'to', 'a', 'new', 'generation',
'of', 'Americans', '--', 'born', 'in', 'this', 'century', ',', 'tempered', 'by', 'war', ',', 'disciplined', 'by', 'a', 'har
d', 'and', 'bitter', 'peace', ',', 'proud', 'of', 'our', 'ancient', 'heritage', '--', 'and', 'unwilling', 'to', 'witness', 'o
r', 'permit', 'the', 'slow', 'undoing', 'of', 'those', 'human', 'rights', 'to', 'which', 'this', 'Nation', 'has', 'always',
'been', 'committed', ',', 'and', 'to', 'which', 'we', 'are', 'committed', 'today', 'at', 'home', 'and', 'around', 'the', 'wor
ld', '.', 'Let', 'every', 'nation', 'know', ',', 'whether', 'it', 'wishes', 'us', 'well', 'or', 'ill', ',', 'that', 'we', 'sh
all', 'pay', 'any', 'price', ',', 'bear', 'any', 'burden', ',', 'meet', 'any', 'hardship', ',', 'support', 'any', 'friend',
',', 'oppose', 'any', 'foe', ',', 'in', 'order', 'to', 'assure', 'the', 'survival', 'and', 'the', 'success', 'of', 'liberty',
```

Fig-2.2.2

c) Nixon

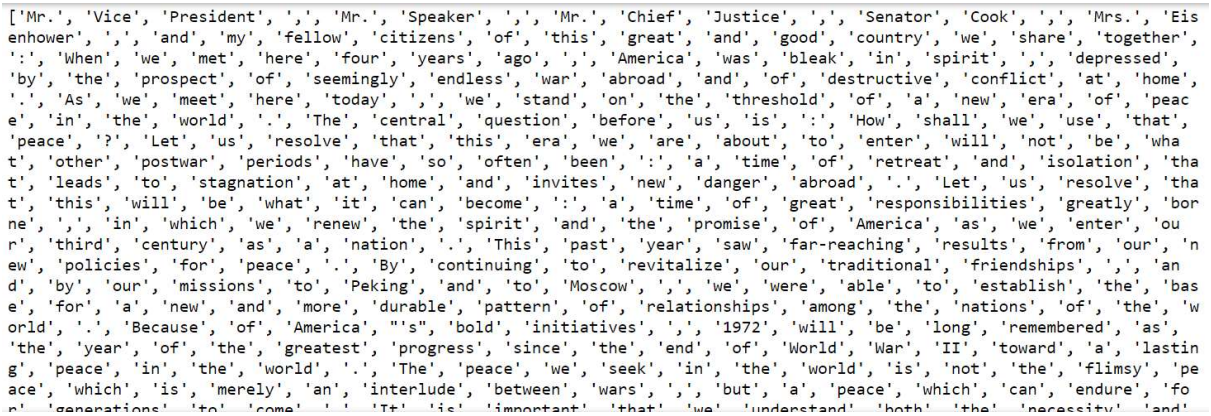

Fig-2.2.3

Conclusion | insight: We would use the library from nltk.corpus import stopwords from nltk.tokenize import word_tokenize. We need these to remove all the English predefined words from each text file separately and with the help of tokenize we would separate each word and remove all the words from the text file.

Q 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)

Conclusion | insight: In three modal looking mostly what time use a particular words, below thee modals all small Snp the now for count of time use words. Command are mention in python file.

a) Roosevelt

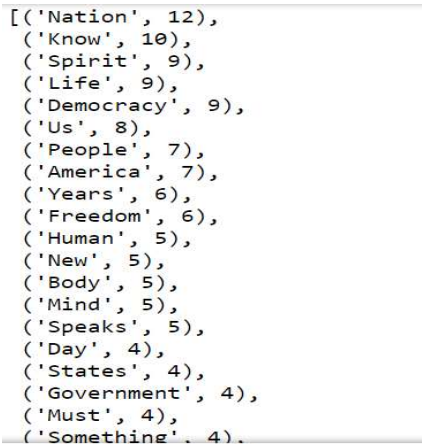

Fig-2.3.1

b) Kennedy

```
[('Let', 16),
 ('Us', 12),
 ('World', 8),
 ('Sides', 8),
 ('New', 7),
 ('Pledge', 7),
 ('Citizens', 5),
 ('Power', 5),
 ('Shall', 5),
 ('Free', 5),
 ('Nations', 5),
 ('Ask', 5),
 ('President', 4),
 ('Fellow', 4),
 ('Freedom', 4),
 ('First', 4),
 ('Americans', 4),
 ('Peace', 4),
 ('Always', 4),
 ('Cannot', 4)
```

Fig-2.3.2

c) Nixon

```
[('Us', 26),
 ('Let', 22),
 ('America', 21),
 ('Peace', 19),
 ('World', 18),
 ('New', 15),
 ('Nation', 11),
 ('Responsibility', 11),
 ('Government', 10),
 ('Great', 9),
 ('Home', 9),
 ('Abroad', 8),
 ('Together', 7),
 ('Years', 7),
 ('Shall', 7),
 ('Policies', 7),
 ('Role', 7),
 ('Make', 7),
 ('Every', 7),
 ('History', 7)
```

Fig-2.3.3

Conclusion | insight: All words are not mention due to more hair shape take .so some small Snp are shown. All three modal command are performed result Snp are provided ntlk are plunk .

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

Conclusion | insight: Using command was mention in python file. Below shown most command using words are show in all three speeches.

a) Roosevelt

Fig-2.4.1

b) Kennedy

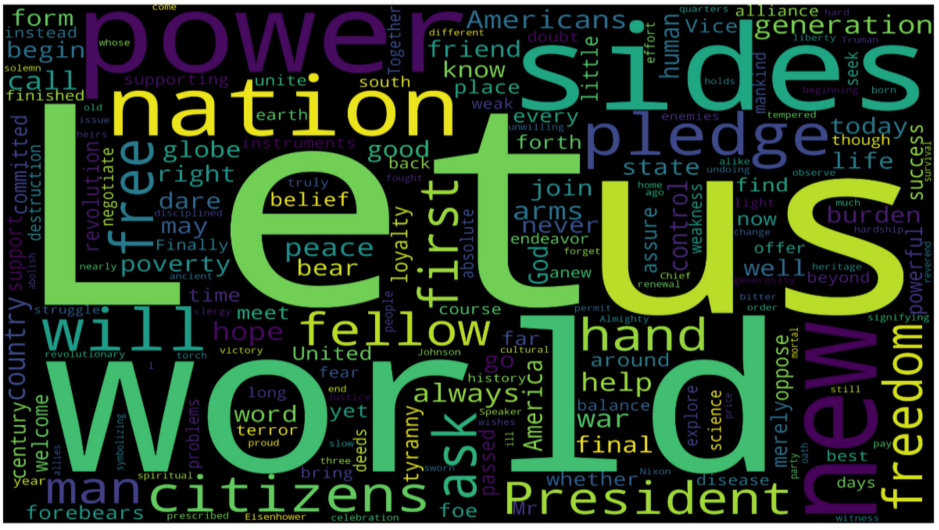

Fig-2.4.2

c) Nixon

Fig-2.4.3

Conclusion | insight: Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. Word clouds are widely used for analysing data from social network websites. Here we are creating the word cloud for Roosevelt speech and we have imported the wordcloud by importing libraries. We are also making sure to remove all the substrings appearing the filtered data.

THANKS