Great Learning

POWER AHEAD

Capstone Project – Project Notes -2 Submission

NAME	DEEPAK SINGH
PROJECT	CUSTOMER CHURN
GROUP	PGP-DSBA
SUBMISSION DATE	4TH FEB 2024

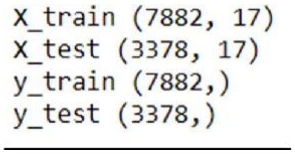
Model Building

In this part of capstone project, we will move towards various model building after EDA and data cleaning performed earlier followed by model tuning and assessing the performance over different metrics Accuracy, F1 Score, Recall, Precession, ROC curve, AUC score, Confusion matrix and classification report. We will choose the model which does not underfit or overfit along with the best accuracy in place.

Splitting Data into Train and Test Dataset: -

Following the accepted market practice, we have divided data into Train and Test datasetinto 70:30 ratio and building various models on training dataset and testing for accuracy over testing dataset.

Below is the shape of Train and Test dataset: -



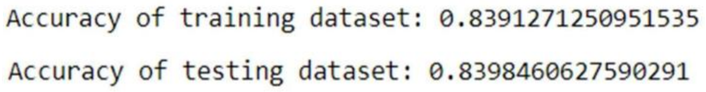
```
X_train (7882, 17)
X_test (3378, 17)
y_train (7882,)
y_test (3378,)
```

Fig 1: - Shape of training and test dataset

Building Logistic Regression Model on Dataset: -

- **Building model with default hyperparameters: -**

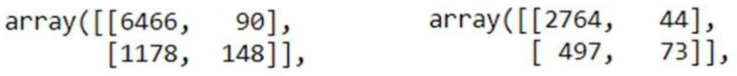
Post splitting data into training and testing data set we fitted logistic regression model into training dataset and performed prediction on training and testing dataset using the same model. We made the first model with default hyperparameters with default solver as Ibfgs. Below are the accuracy scores obtained from this model: -



```
Accuracy of training dataset: 0.8391271250951535

Accuracy of testing dataset: 0.8398460627590291
```

Fig 2: - Accuracy from Logistic Regression

Below is the confusion matrix obtained from this model: -

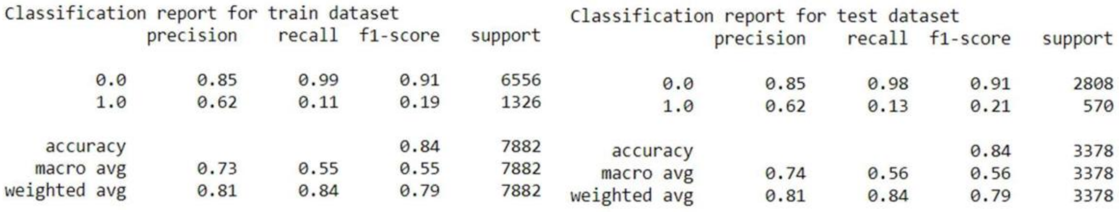


```
array([[6466,   90],         array([[2764,   44],
       [1178,  148]],               [ 497,   73]],
```

Train **Test**

Fig 3: - Confusion Matrix from Logistic Regression

Below is the classification report obtained from this model: -

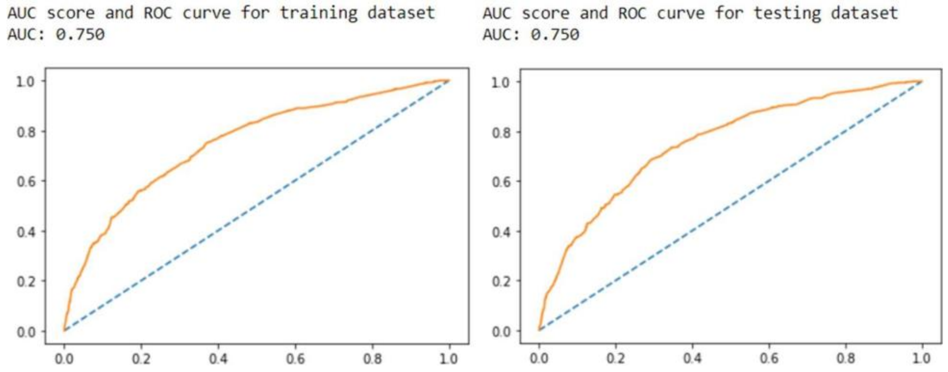


```
Classification report for train dataset
              precision    recall  f1-score   support

         0.0       0.85      0.99      0.91      6556
         1.0       0.62      0.11      0.19      1326

    accuracy                           0.84      7882
   macro avg       0.73      0.55      0.55      7882
weighted avg       0.81      0.84      0.79      7882
```

```
Classification report for test dataset
              precision    recall  f1-score   support

         0.0       0.85      0.98      0.91      2808
         1.0       0.62      0.13      0.21       570

    accuracy                           0.84      3378
   macro avg       0.74      0.56      0.56      3378
weighted avg       0.81      0.84      0.79      3378
```

Fig 4: - Classification Report from Logistic Regression

Below is the AUC score and ROC curve obtained from this model: -

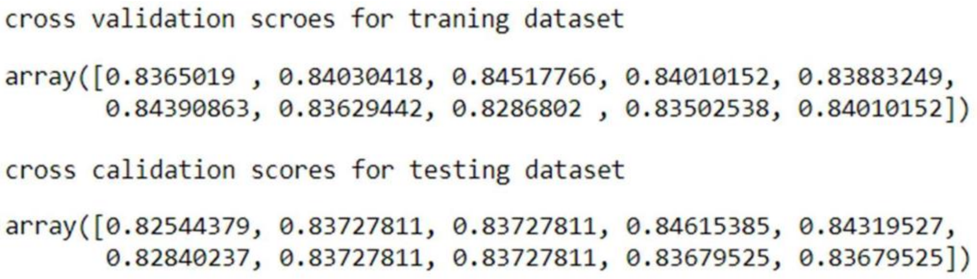


```
AUC score and ROC curve for training dataset
AUC: 0.750
```

```
AUC score and ROC curve for testing dataset
AUC: 0.750
```

Fig 5: - ROC Curve & AUC Score From Logistic Regression

Below are the 10-fold cross validation for logistic regression with default values: -
Cross-validation is a process to check if the built model is correct or not. Below are the 10-fold cross validation scores: -



```
cross validation scroes for traning dataset

array([0.8365019 , 0.84030418, 0.84517766, 0.84010152, 0.83883249,
       0.84390863, 0.83629442, 0.8286802 , 0.83502538, 0.84010152])

cross calidation scores for testing dataset

array([0.82544379, 0.83727811, 0.83727811, 0.84615385, 0.84319527,
       0.82840237, 0.83727811, 0.83727811, 0.83679525, 0.83679525])
```

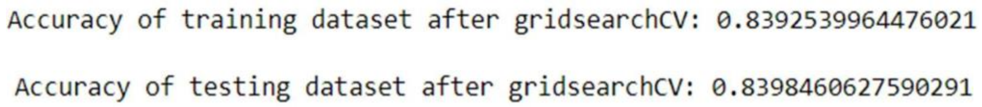
Fig 6: - Cross Validation Scores From Logistic Regression

We can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

<u>Building model using GridSearchCV and analyzing the best parameters: -</u>

We use GridSearchCV to find an optimal combination of hyperparameters that minimizes a predefined loss function to give better results. Performed GridSearchCV with various hyper-parameters like “solver”, “penalty” and “tol” and we can find that “ibfgs” solver along with “none” penalty worked as best parameters for this dataset. However, we have also observed that the difference in accuracy for train and test dataset is very marginal and not much of significant.

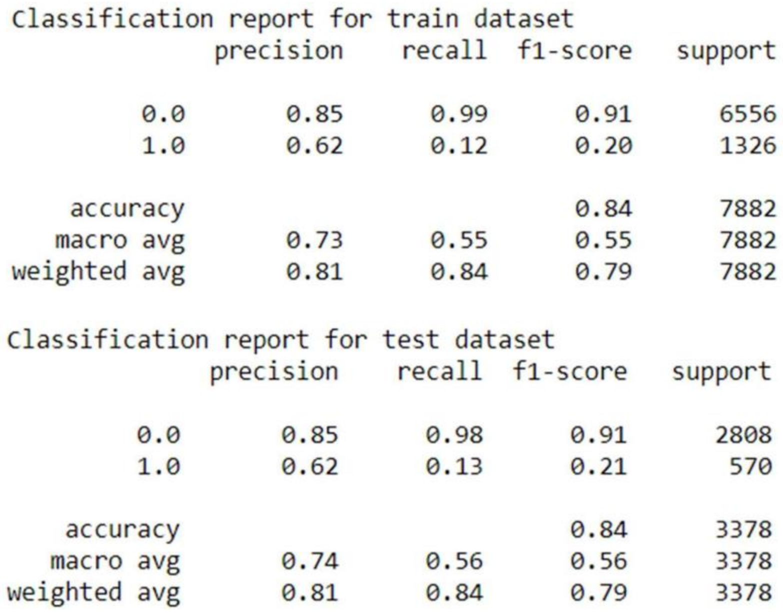
Below are the accuracy scores obtained from this model using GridSearchCV: -



```
Accuracy of training dataset after gridsearchCV: 0.8392539964476021

Accuracy of testing dataset after gridsearchCV: 0.8398460627590291
```

Fig 7: - Accuracy from Logistic Regression using hyper-parameter

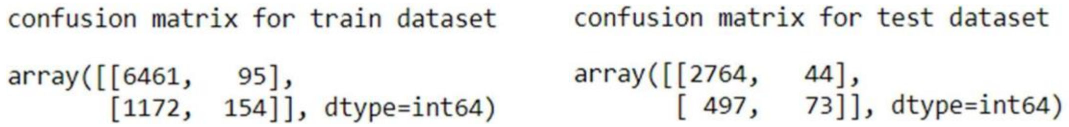
Below is the classification report obtained from this model using GridSearchCV: -



```
Classification report for train dataset
              precision    recall  f1-score   support

         0.0       0.85      0.99      0.91      6556
         1.0       0.62      0.12      0.20      1326

    accuracy                           0.84      7882
   macro avg       0.73      0.55      0.55      7882
weighted avg       0.81      0.84      0.79      7882

Classification report for test dataset
              precision    recall  f1-score   support

         0.0       0.85      0.98      0.91      2808
         1.0       0.62      0.13      0.21       570

    accuracy                           0.84      3378
   macro avg       0.74      0.56      0.56      3378
weighted avg       0.81      0.84      0.79      3378
```

Fig 8: - Classification Report from Logistic Regression using hyper-parameter

Below is the confusion matrix obtained from this model using GridSearchCV: -



```
confusion matrix for train dataset        confusion matrix for test dataset

array([[6461,   95],                      array([[2764,   44],
       [1172,  154]], dtype=int64)               [ 497,   73]], dtype=int64)
```

Fig 9: - Confusion Matrix from Logistic Regression using hyper-parameter

Below is the AUC score and ROC curve obtained from this model using GridSearchCV: -

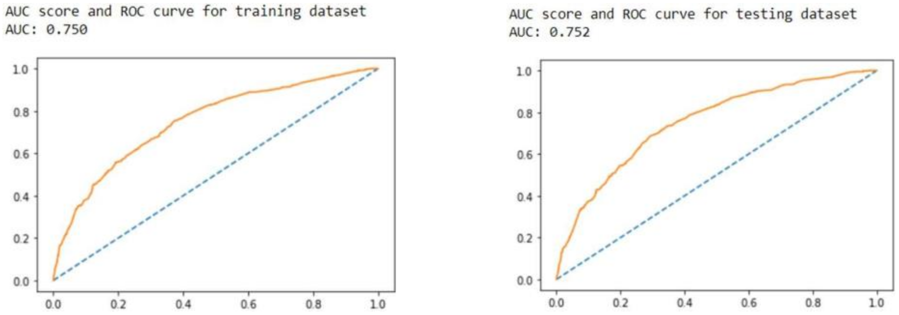
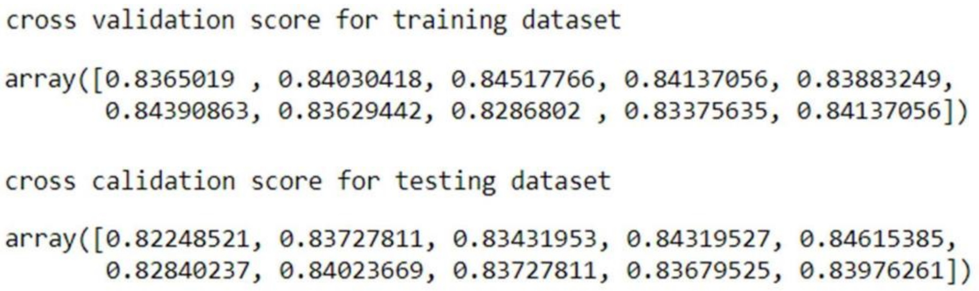


Fig 10: - ROC Curve and AUC Score from Logistic Regression using hyper-parameter

Below are the 10-fold cross validation scores: -



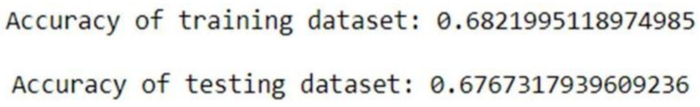
```
cross validation score for training dataset

array([0.8365019 , 0.84030418, 0.84517766, 0.84137056, 0.83883249,
       0.84390863, 0.83629442, 0.8286802 , 0.83375635, 0.84137056])

cross calidation score for testing dataset

array([0.82248521, 0.83727811, 0.83431953, 0.84319527, 0.84615385,
       0.82840237, 0.84023669, 0.83727811, 0.83679525, 0.83976261])
```

Fig 10: - Cross Validation Scores From Logistic Regression using hyper-parameter

Building model using SMOTE: -

In our previous analysis we have seen that the data is imbalanced in nature. We can use SMOTE technique to balance the data and then tried building model on the balanced data to check if we can see some significant improvement in accuracy for training and testing dataset. After building model on balanced dataset and checking on accuracy we can see that the performance is not that significant in terms of accuracy.

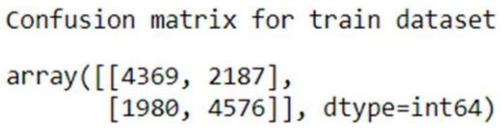
Below are the accuracy scores obtained from balanced data: -



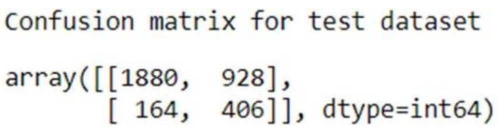
```
Accuracy of training dataset: 0.6821995118974985

Accuracy of testing dataset: 0.6767317939609236
```

Fig 11: - Accuracy Score from Logistic Regression with SMOTE

Below is the confusion matrix obtained from balanced data: -



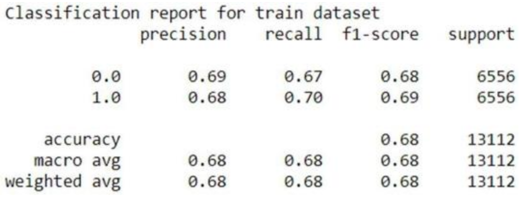
```
Confusion matrix for train dataset

array([[4369, 2187],
       [1980, 4576]], dtype=int64)
```



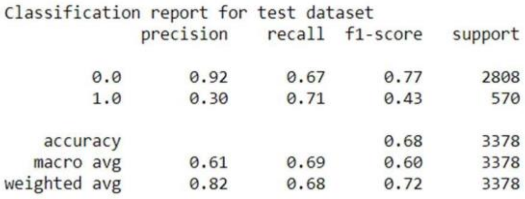
```
Confusion matrix for test dataset

array([[1880,  928],
       [ 164,  406]], dtype=int64)
```

Fig 12: - Confusion Matrix from Logistic Regression with SMOTE

Below is the classification report obtained from balanced data: -



```
Classification report for train dataset
              precision    recall  f1-score   support

         0.0       0.69      0.67      0.68      6556
         1.0       0.68      0.70      0.69      6556

    accuracy                           0.68     13112
   macro avg       0.68      0.68      0.68     13112
weighted avg       0.68      0.68      0.68     13112
```



```
Classification report for test dataset
              precision    recall  f1-score   support

         0.0       0.92      0.67      0.77      2808
         1.0       0.30      0.71      0.43       570

    accuracy                           0.68      3378
   macro avg       0.61      0.69      0.60      3378
weighted avg       0.82      0.68      0.72      3378
```

Fig 13: - Classification Report from Logistic Regression with SMOTE

Below are the AUC scores and ROC curve obtained from balanced data: -

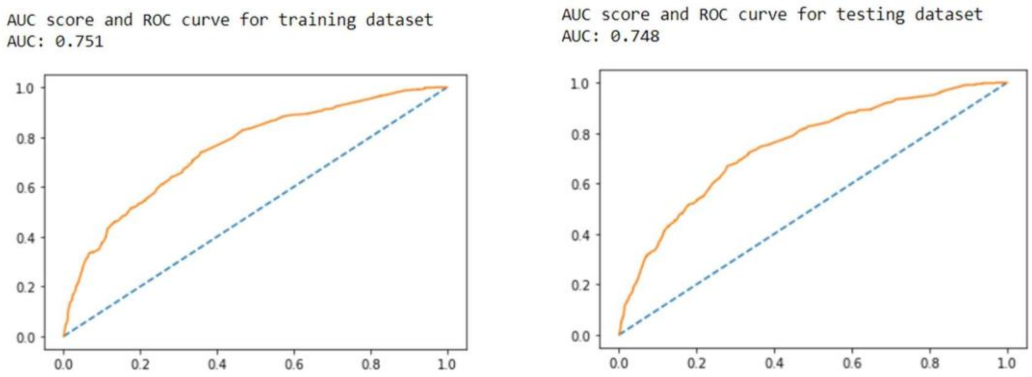
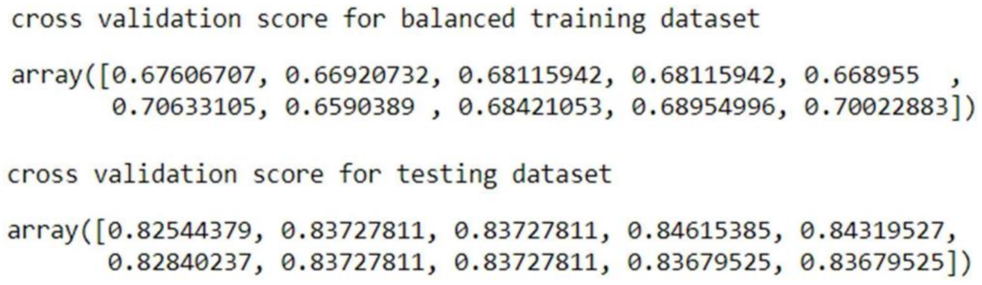


Fig 14: - ROC Curve & AUC Scores From Logistic Regression with SMOTE

Below are the 10-fold cross validation scores: -



```
cross validation score for balanced training dataset

array([0.67606707, 0.66920732, 0.68115942, 0.68115942, 0.668955  ,
       0.70633105, 0.6590389 , 0.68421053, 0.68954996, 0.70022883])

cross validation score for testing dataset

array([0.82544379, 0.83727811, 0.83727811, 0.84615385, 0.84319527,
       0.82840237, 0.83727811, 0.83727811, 0.83679525, 0.83679525])
```

Fig 15: - Cross Validation Scores From Logistic Regression with SMOTE

We can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

Inference/Conclusion from Logistic Regression Model: -

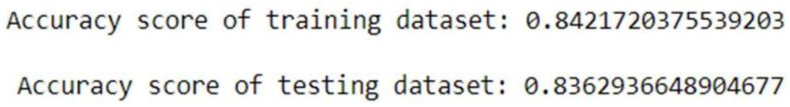
From the above we can conclude that the data is neither “Overfit” nor “Underfit” in nature. And we can also inference that the model built using GridSearchCV is best optimized considering the best parameters obtained. However, the accuracy scores along with recall, precision, F1 values, ROC curve and AUC score are not that significant as compared with models built with default values and balanced data (SMOTE). Model built on balanced data set using SMOTE technique works well in training dataset however we can see a significant deplete in accuracy when it comes to testing dataset.

Building Linear Discriminant Analysis Model (LDA)

Building model with default hyperparameters: -

From the above split into training and testing dataset we are building Linear Discriminant Analysis (LDA) model to check it this can outperform Logistic regression model and we can choose form best for further predictions. Firstly, we are building model using default values of LDA. That is, solver as“svd” and shrinkage as “none”.

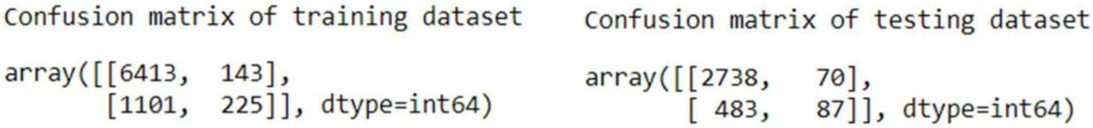
Below are the accuracy scores obtained from this model: -



```
Accuracy score of training dataset: 0.8421720375539203

 Accuracy score of testing dataset: 0.8362936648904677
```

Fig 16: - Accuracy from LDA

Below is the confusion matrix obtained from this model: -

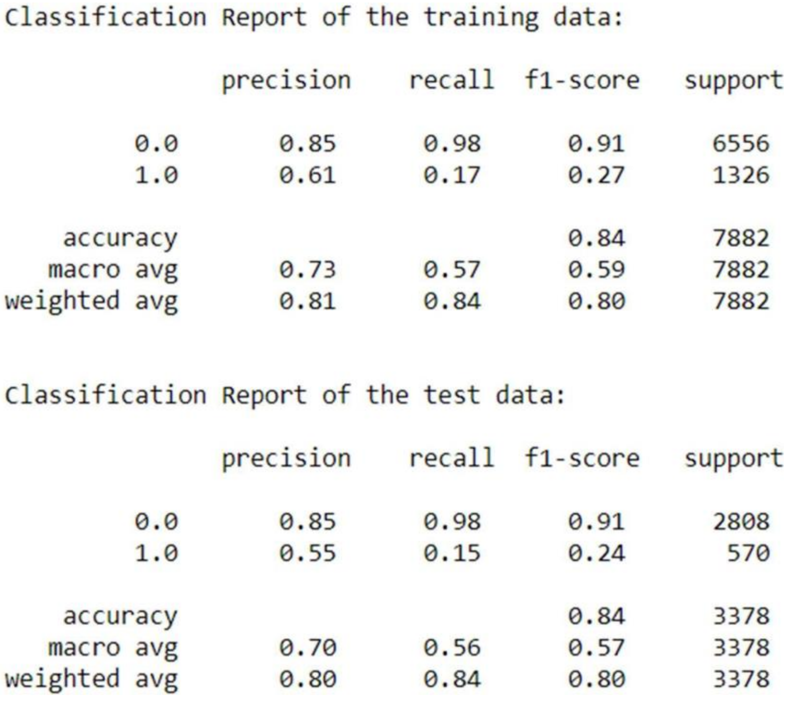


```
Confusion matrix of training dataset

array([[6413,  143],
       [1101,  225]], dtype=int64)
```

```
Confusion matrix of testing dataset

array([[2738,   70],
       [ 483,   87]], dtype=int64)
```

Fig 17: - Confusion Matrix from LDA

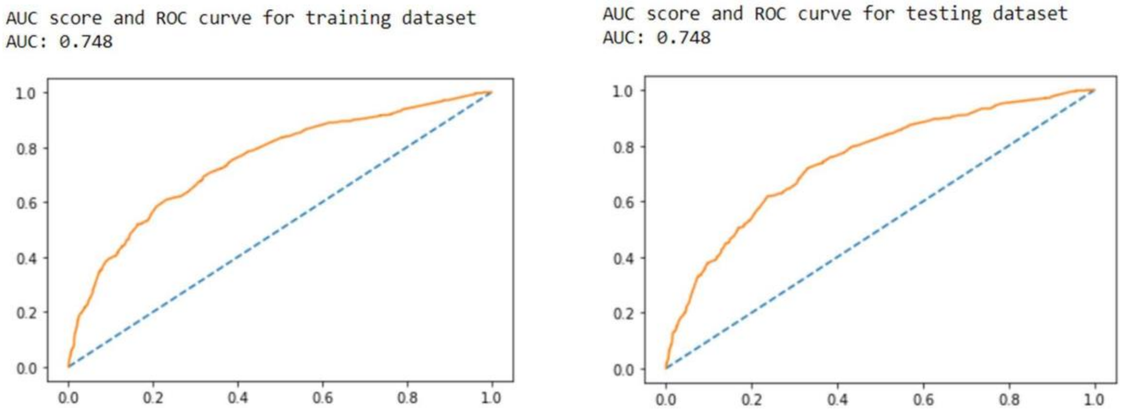
Below is the classification report obtained from this model: -



```
Classification Report of the training data:

              precision    recall  f1-score   support

         0.0       0.85      0.98      0.91      6556
         1.0       0.61      0.17      0.27      1326

    accuracy                           0.84      7882
   macro avg       0.73      0.57      0.59      7882
weighted avg       0.81      0.84      0.80      7882


Classification Report of the test data:

              precision    recall  f1-score   support

         0.0       0.85      0.98      0.91      2808
         1.0       0.55      0.15      0.24       570

    accuracy                           0.84      3378
   macro avg       0.70      0.56      0.57      3378
weighted avg       0.80      0.84      0.80      3378
```

Fig 18: - Classification Report from LDA

Below are the AUC scores and ROC curves obtained from this model: -

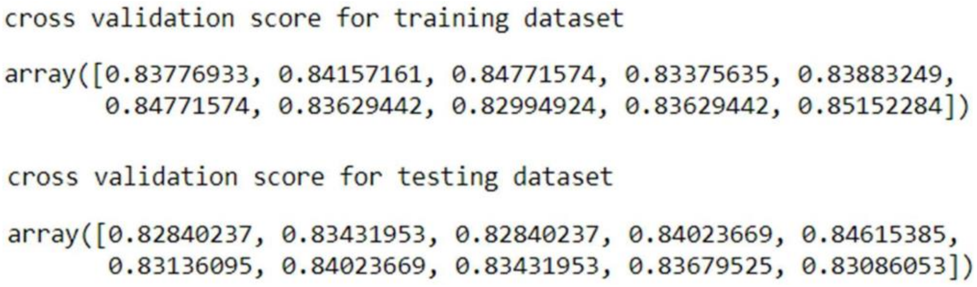


```
AUC score and ROC curve for training dataset
AUC: 0.748
```

```
AUC score and ROC curve for testing dataset
AUC: 0.748
```

Fig 19: - ROC Curve and AUC Score From LDA

Below are the 10-fold cross validation scores: -



```
cross validation score for training dataset

array([0.83776933, 0.84157161, 0.84771574, 0.83375635, 0.83883249,
       0.84771574, 0.83629442, 0.82994924, 0.83629442, 0.85152284])

cross validation score for testing dataset

array([0.82840237, 0.83431953, 0.82840237, 0.84023669, 0.84615385,
       0.83136095, 0.84023669, 0.83431953, 0.83679525, 0.83086053])
```

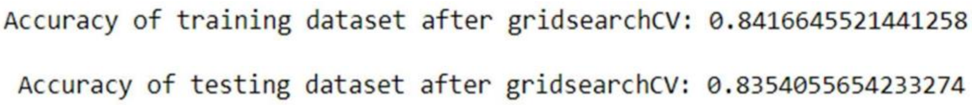
Fig 20: - Cross Validation Score from LDA

We can observe that the cross validations scores are almost same for all the folds. Which indicates thatthe model built is correct.

Building model using GridSearchCV and analyzing the best parameters: -

Using GridSearchCV function we tried finding the best parameters to further tune-in the above model for better accuracy and we have find that shrinkage as “auto”, solver as “lsqr” and tol value of “0.001” gives the best model considering accuracy, precision, recall, F1, ROC curve and AUC score.

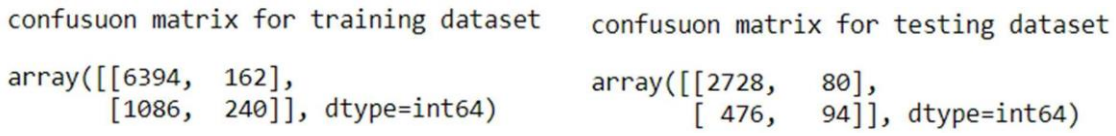
Below are the accuracy scores obtained from model built using GridSearchCV: -



```
Accuracy of training dataset after gridsearchCV: 0.8416645521441258

 Accuracy of testing dataset after gridsearchCV: 0.8354055654233274
```

Fig 21: - Accuracy Score from LDA with Hypertuning

Below is the confusion matrix obtained from model built using GridSearchCV: -

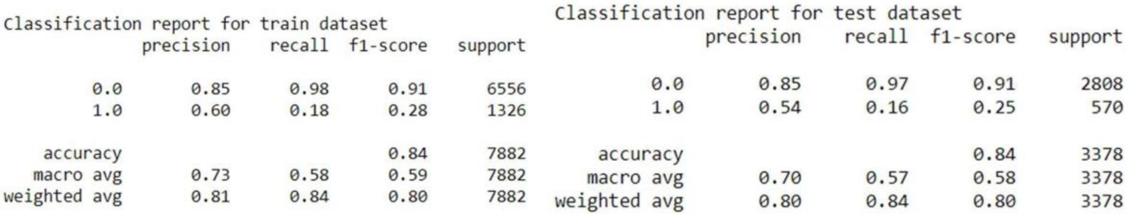


```
confusuon matrix for training dataset

array([[6394,  162],
       [1086,  240]], dtype=int64)
```

```
confusuon matrix for testing dataset

array([[2728,   80],
       [ 476,   94]], dtype=int64)
```

Fig 22: - Confusion Matrix from LDA with Hypertuning

Below are the classification reports obtained from model built using GridSearchCV: -

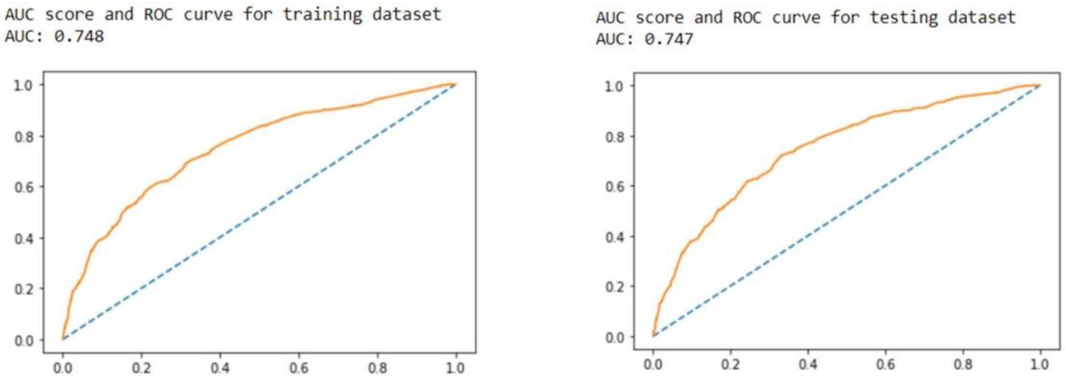


```
Classification report for train dataset
              precision    recall  f1-score   support

         0.0       0.85      0.98      0.91      6556
         1.0       0.60      0.18      0.28      1326

    accuracy                           0.84      7882
   macro avg       0.73      0.58      0.59      7882
weighted avg       0.81      0.84      0.80      7882
```

```
Classification report for test dataset
              precision    recall  f1-score   support

         0.0       0.85      0.97      0.91      2808
         1.0       0.54      0.16      0.25       570

    accuracy                           0.84      3378
   macro avg       0.70      0.57      0.58      3378
weighted avg       0.80      0.84      0.80      3378
```

Fig 23: - Classification Report from LDA with Hypertuning

Below are the ROC curve and AUC scores obtained from model built using GridSearchCV: -

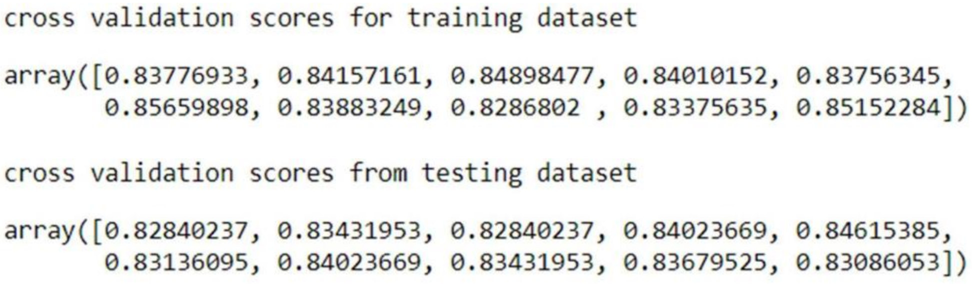


```
AUC score and ROC curve for training dataset
AUC: 0.748
```

```
AUC score and ROC curve for testing dataset
AUC: 0.747
```

Fig 24: - ROC Curve and AUC Score From LDA with Hypertuning

Below are the 10-fold cross validation scores: -



```
cross validation scores for training dataset

array([0.83776933, 0.84157161, 0.84898477, 0.84010152, 0.83756345,
       0.85659898, 0.83883249, 0.8286802 , 0.83375635, 0.85152284])

cross validation scores from testing dataset

array([0.82840237, 0.83431953, 0.82840237, 0.84023669, 0.84615385,
       0.83136095, 0.84023669, 0.83431953, 0.83679525, 0.83086053])
```

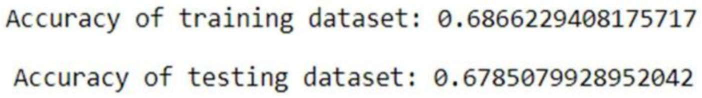
Fig 71: - Cross Validation Scores From LDA with Hypertuning

We can observe that the cross validations scores are almost same for all the folds. Which indicates thatthe model built is correct.

Building LDA model using SMOTE: -

From above descriptive analysis we can conclude that the original data provided is imbalance in nature and by using SMOTE technique we can balance the data to check if the model can outperform when data is balanced. We have applied SMOTE technique to oversample the data and to obtain a balanced dataset.

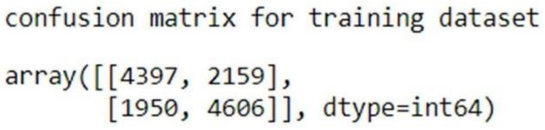
Below are the accuracy scores obtained from balanced dataset: -



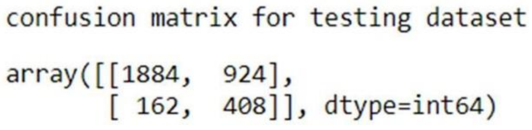
```
Accuracy of training dataset: 0.6866229408175717

Accuracy of testing dataset: 0.6785079928952042
```

Fig 25: - Accuracy from LDA with SMOTE

Below is the confusion matrix obtained from balanced dataset: -



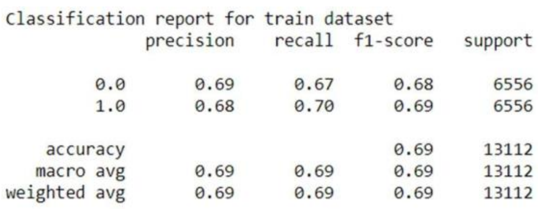
```
confusion matrix for training dataset

array([[4397, 2159],
       [1950, 4606]], dtype=int64)
```



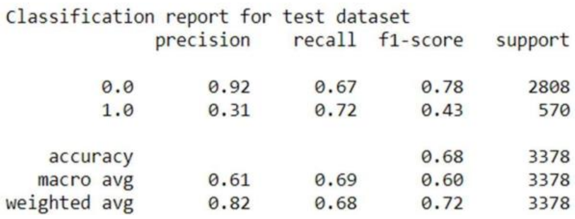
```
confusion matrix for testing dataset

array([[1884,  924],
       [ 162,  408]], dtype=int64)
```

Fig 26: - Confusion Matrix from LDA with SMOTE

Below is the classification report obtained from balanced dataset: -



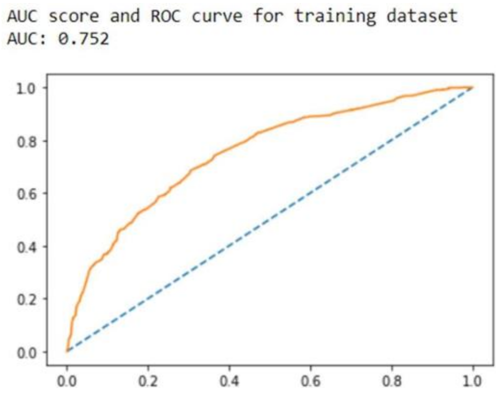
```
Classification report for train dataset
              precision    recall  f1-score   support

         0.0       0.69      0.67      0.68      6556
         1.0       0.68      0.70      0.69      6556

    accuracy                           0.69     13112
   macro avg       0.69      0.69      0.69     13112
weighted avg       0.69      0.69      0.69     13112
```



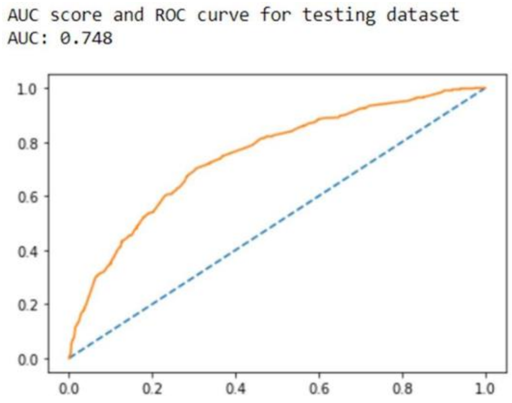
```
Classification report for test dataset
              precision    recall  f1-score   support

         0.0       0.92      0.67      0.78      2808
         1.0       0.31      0.72      0.43       570

    accuracy                           0.68      3378
   macro avg       0.61      0.69      0.60      3378
weighted avg       0.82      0.68      0.72      3378
```

Fig 27: - Classification Report from LDA with SMOTE

Below are the ROC curve and AUC scores obtained from balanced dataset: -



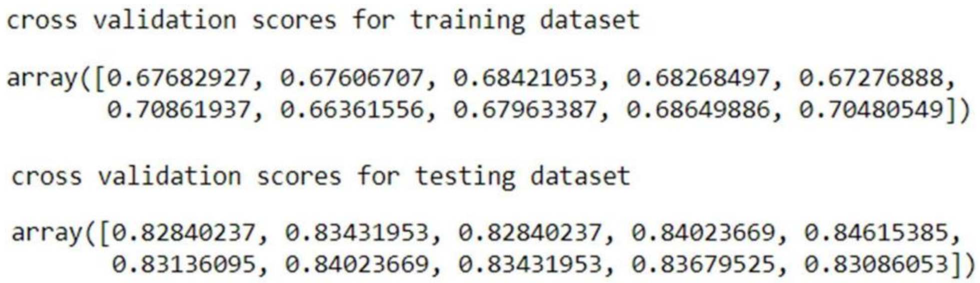
AUC score and ROC curve for training dataset
AUC: 0.752



AUC score and ROC curve for testing dataset
AUC: 0.748

Fig 28: - ROC Curve and AUC Score From LDA with SMOTE

Below are the 10-fold cross validation scores: -



```
cross validation scores for training dataset

array([0.67682927, 0.67606707, 0.68421053, 0.68268497, 0.67276888,
       0.70861937, 0.66361556, 0.67963387, 0.68649886, 0.70480549])
```

```
cross validation scores for testing dataset

array([0.82840237, 0.83431953, 0.82840237, 0.84023669, 0.84615385,
       0.83136095, 0.84023669, 0.83431953, 0.83679525, 0.83086053])
```

Fig 29: - Cross Validation Scores From LDA with SMOTE

We can observe that the cross validations scores are almost same for all the folds. Which indicates thatthe model built is correct.

Inference/Conclusion from LDA Model: -

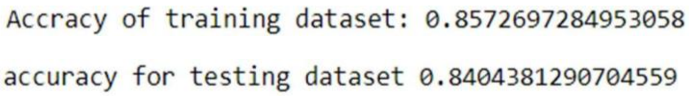
From the above we can conclude that the data is neither “Overfit” nor “Underfit” in nature. And we can also inference that the model built using the parameters from GridSearchCV is best optimized. However, the accuracy scores along with recall, precision, F1 values, ROC curve and AUC score are not that significant as compared with models built with default values and model built using GridSearchCV. Model built on balance data set using SMOTE technique works well in training dataset however we can see a significant deplete in accuracy when it comes to testing dataset.

BUILDING KNN MODEL: -

Building model with default hyperparameters: -

Post splitting data into training and testing data set we fitted KNN model into training dataset and performed prediction on training and testing dataset using the same model. We made the first model withdefault hyperparameters with default value of n_neighbour as “5”.

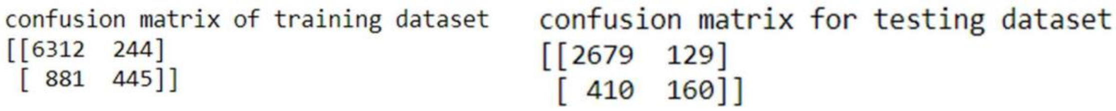
Below are the accuracy scores obtained from this model: -



```
Accracy of training dataset: 0.8572697284953058

accuracy for testing dataset 0.8404381290704559
```

Fig 30: - Accuracy Scores From KNN

Below are the confusion matrices obtained from this model: -

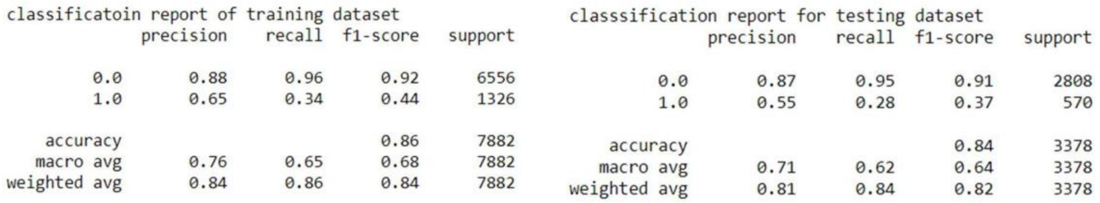


```
confusion matrix of training dataset
[[6312  244]
 [ 881  445]]
```

```
confusion matrix for testing dataset
[[2679  129]
 [ 410  160]]
```

Fig 31: - Confusion Matrix from KNN

Below are the classification Report obtained from this model: -

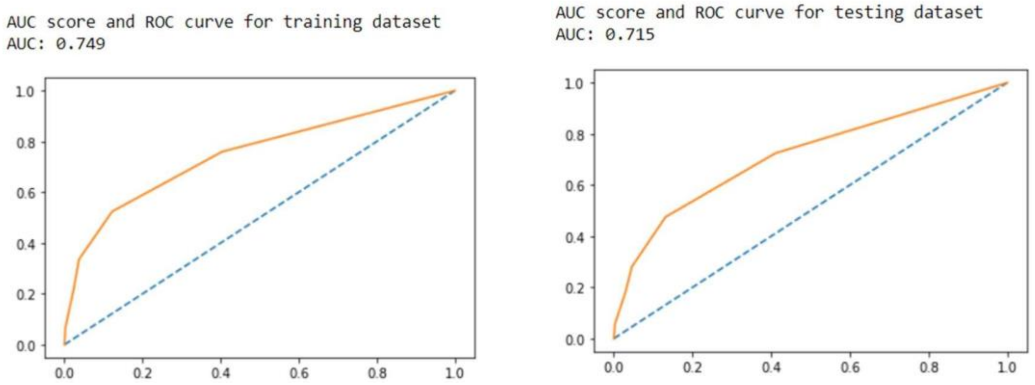


```
classificatoin report of training dataset
              precision    recall  f1-score   support

         0.0       0.88      0.96      0.92      6556
         1.0       0.65      0.34      0.44      1326

    accuracy                           0.86      7882
   macro avg       0.76      0.65      0.68      7882
weighted avg       0.84      0.86      0.84      7882
```

```
classsification report for testing dataset
              precision    recall  f1-score   support

         0.0       0.87      0.95      0.91      2808
         1.0       0.55      0.28      0.37       570

    accuracy                           0.84      3378
   macro avg       0.71      0.62      0.64      3378
weighted avg       0.81      0.84      0.82      3378
```

Fig 32: - Classification Report from KNN

Below are the AUC scores and ROC curves obtained from this model: -

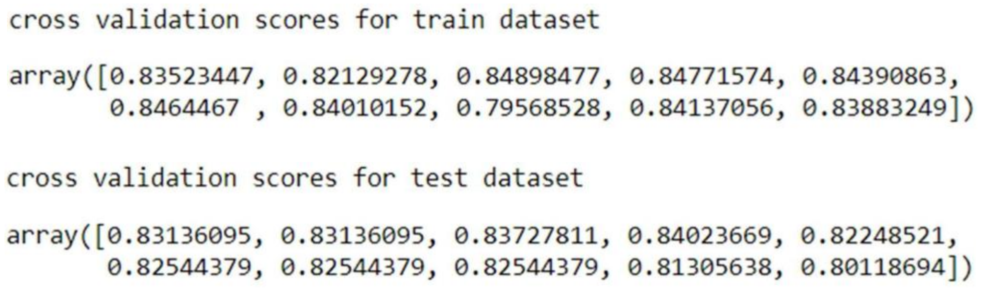


```
AUC score and ROC curve for training dataset
AUC: 0.749
```

```
AUC score and ROC curve for testing dataset
AUC: 0.715
```

Fig 33: - ROC Curve and AUC Scores From KNN

Below are the 10-fold cross validation scores: -



```
cross validation scores for train dataset

array([0.83523447, 0.82129278, 0.84898477, 0.84771574, 0.84390863,
       0.8464467 , 0.84010152, 0.79568528, 0.84137056, 0.83883249])

cross validation scores for test dataset

array([0.83136095, 0.83136095, 0.83727811, 0.84023669, 0.82248521,
       0.82544379, 0.82544379, 0.82544379, 0.81305638, 0.80118694])
```

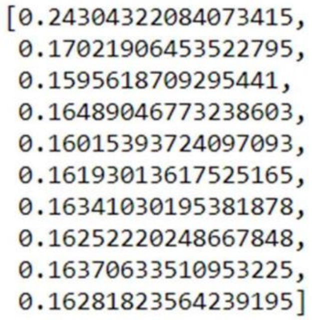
Fig 34: - Cross Validation Scores From KNN

We can observe that the cross validations scores are almost same for all the folds. Which indicates thatthe model built is correct.

Find the right value of n_neighbor: -

It’s very important to have the right value of n_neighbors to fetch the best accuracy from the model. We can decide on the best value for n_neighbors based on MSE (mean squared error) scores. The value with least score of MSE indicated least error and will fetch the best optimized n_neighbors value.

Below are the MSE scores: -



```
[0.24304322084073415,
 0.17021906453522795,
 0.1595618709295441,
 0.16489046773238603,
 0.16015393724097093,
 0.16193013617525165,
 0.16341030195381878,
 0.16252220248667848,
 0.16370633510953225,
 0.16281823564239195]
```

Fig 35: - MSE Scores

Below is the graphical version of MSE scores across numerous values of n_neighbors.

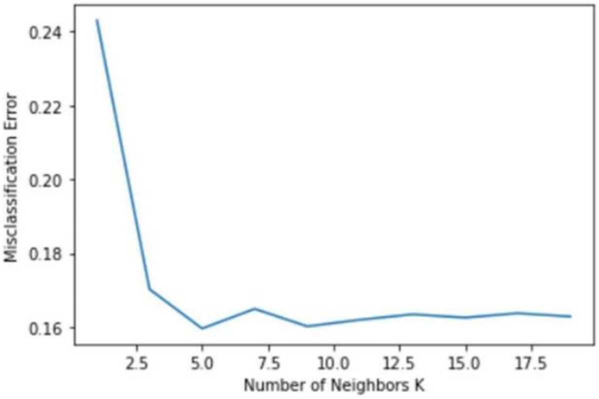


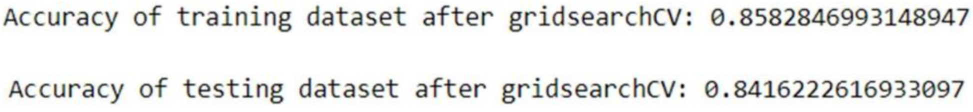
Fig 36: - Graphical Version of MSE Score

From the above plotted graph we can see the n_neighbors with value "5" gives the least MSE score. With which we can proceed and build KNN model with n_neighbor value as "5" which is also the default n_neighbor. **Hence, different model building with correct number of n_neighbor is not required as it's the same as default value if n_neighbor.**

Building model using GridSearchCV and getting the best hyperparameters: -

After building the model with its default values as shown above, we will try and find the best hyper parameters to check if we can outperform the accuracy achieved by the model built with default values of hyperparameter. From GridSearchCV we found that the best parameters are "ball-tree" asalgorithm, "Manhattan" as metrics, "5" as n_neighbors and "distance" as weights.

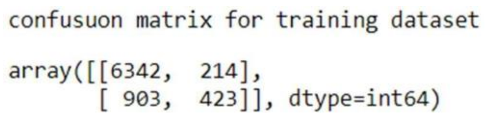
Below are the accuracy scores obtained from this model using GridSearchCV: -



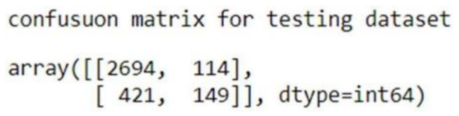
```
Accuracy of training dataset after gridsearchCV: 0.8582846993148947

Accuracy of testing dataset after gridsearchCV: 0.8416222616933097
```

Fig 37: - Accuracy from KNN with Hyperparamter Tuning

Below are the confusion matrices obtained from this model using GridSearchCV: -



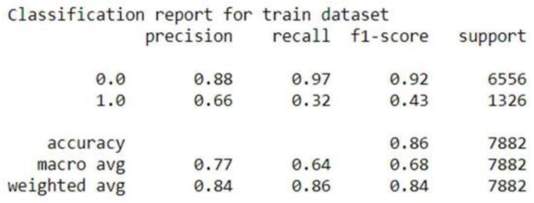
```
confusuon matrix for training dataset

array([[6342,  214],
       [ 903,  423]], dtype=int64)
```



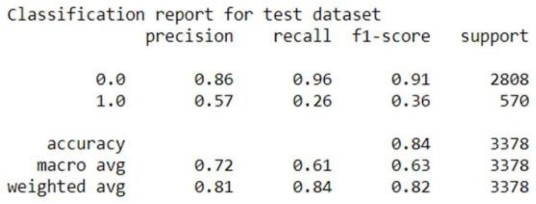
```
confusuon matrix for testing dataset

array([[2694,  114],
       [ 421,  149]], dtype=int64)
```

Fig 38: - Confusion Matrix from KNN with Hyperparamter Tuning

Below is the classification report obtained from this model using GridSearchCV: -



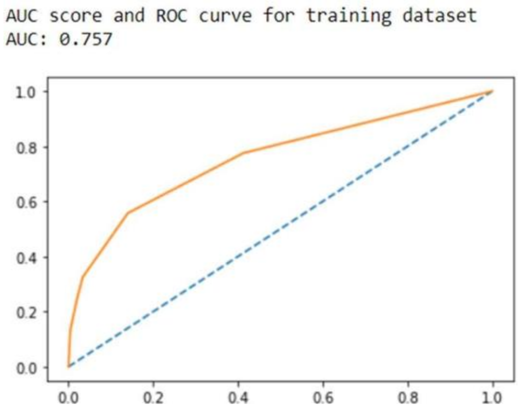
```
Classification report for train dataset
              precision    recall  f1-score   support

         0.0       0.88      0.97      0.92      6556
         1.0       0.66      0.32      0.43      1326

    accuracy                           0.86      7882
   macro avg       0.77      0.64      0.68      7882
weighted avg       0.84      0.86      0.84      7882
```



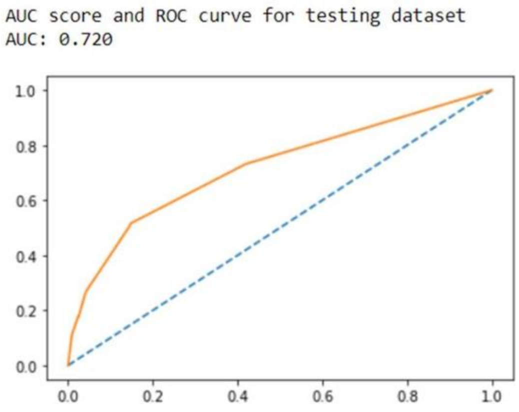
```
Classification report for test dataset
              precision    recall  f1-score   support

         0.0       0.86      0.96      0.91      2808
         1.0       0.57      0.26      0.36       570

    accuracy                           0.84      3378
   macro avg       0.72      0.61      0.63      3378
weighted avg       0.81      0.84      0.82      3378
```

Fig 39: - Classification Report from KNN with Hyperparamter Tuning

Below are the AUC scores and ROC curves obtained from this model using GridSearchCV: -



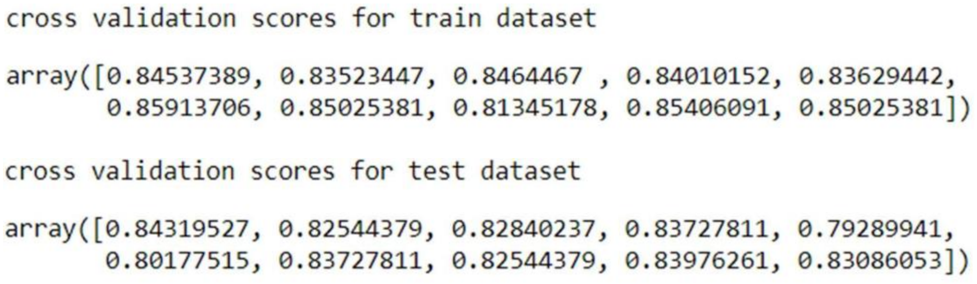
```
AUC score and ROC curve for training dataset
AUC: 0.757
```



```
AUC score and ROC curve for testing dataset
AUC: 0.720
```

Fig 40: - ROC Curve and AUC Score From KNN with Hyperparamter Tuning

Below are the 10-fold cross validation scores: -



```
cross validation scores for train dataset

array([0.84537389, 0.83523447, 0.8464467 , 0.84010152, 0.83629442,
       0.85913706, 0.85025381, 0.81345178, 0.85406091, 0.85025381])

cross validation scores for test dataset

array([0.84319527, 0.82544379, 0.82840237, 0.83727811, 0.79289941,
       0.80177515, 0.83727811, 0.82544379, 0.83976261, 0.83086053])
```

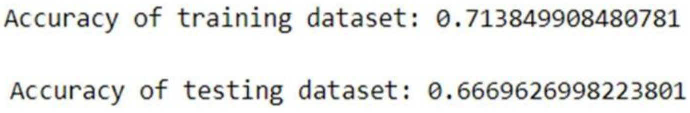
Fig 41: Cross Validation Scores from KNN with Hyperparamter Tuning

We can observe that the cross validations scores are almost same for all the folds. Which indicates thatthe model built is correct.

Building model using SMOTE: -

From above descriptive analysis we can conclude that the original data provided is imbalance in nature and by using SMOTE technique we can balance the data to check if the model can outperform when data is balanced. We have applied SMOTE technique to oversample the data and to obtain a balanced dataset.

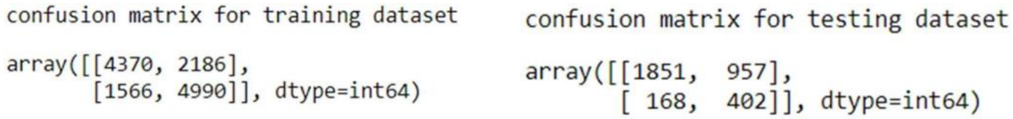
Below are the accuracy scores obtained from balanced dataset: -



```
Accuracy of training dataset: 0.713849908480781

Accuracy of testing dataset: 0.6669626998223801
```

Fig 42: Accuracy Score from KNN with SMOTE

Below are the confusion matrices obtained from balanced dataset: -

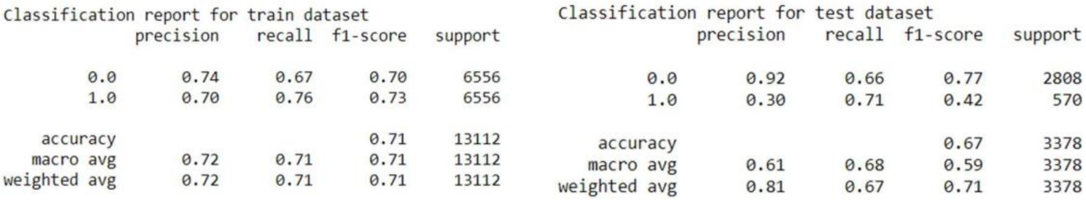


```
confusion matrix for training dataset

array([[4370, 2186],
       [1566, 4990]], dtype=int64)
```

```
confusion matrix for testing dataset

array([[1851,  957],
       [ 168,  402]], dtype=int64)
```

Fig 43: Confusion Matrix from KNN with SMOTE

Below are the classification reports obtained from balanced dataset: -

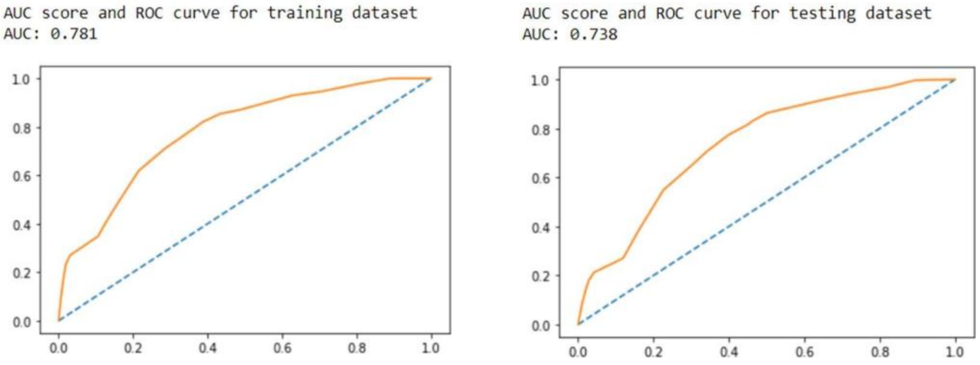


```
Classification report for train dataset
              precision    recall  f1-score   support

         0.0       0.74      0.67      0.70      6556
         1.0       0.70      0.76      0.73      6556

    accuracy                           0.71     13112
   macro avg       0.72      0.71      0.71     13112
weighted avg       0.72      0.71      0.71     13112
```

```
Classification report for test dataset
              precision    recall  f1-score   support

         0.0       0.92      0.66      0.77      2808
         1.0       0.30      0.71      0.42       570

    accuracy                           0.67      3378
   macro avg       0.61      0.68      0.59      3378
weighted avg       0.81      0.67      0.71      3378
```

Fig 44: Classification Reports from KNN with SMOTE

Below are the AUC scores and ROC curves obtained from balanced dataset: -

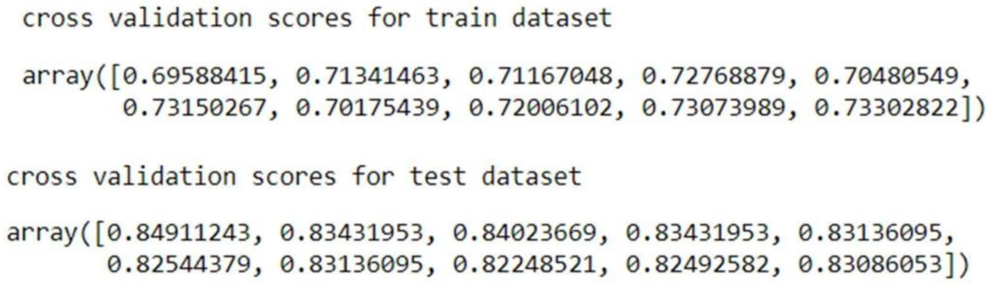


```
AUC score and ROC curve for training dataset
AUC: 0.781
```

```
AUC score and ROC curve for testing dataset
AUC: 0.738
```

Fig 45: ROC Curve and AUC Scores from KNN with SMOTE

Below are the 10-fold cross validation scores: -



```
 cross validation scores for train dataset

 array([0.69588415, 0.71341463, 0.71167048, 0.72768879, 0.70480549,
        0.73150267, 0.70175439, 0.72006102, 0.73073989, 0.73302822])

cross validation scores for test dataset

array([0.84911243, 0.83431953, 0.84023669, 0.83431953, 0.83136095,
       0.82544379, 0.83136095, 0.82248521, 0.82492582, 0.83086053])
```

Fig 46: Cross Validation Scores From KNN with SMOTE

We can observe that the cross validations scores are almost same for all the folds. Which indicates that the model built is correct.

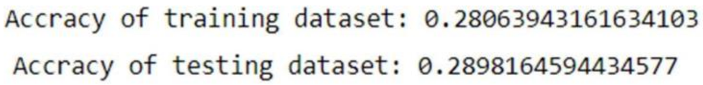
Inference/Conclusion from KNN Model: -

From the above we can conclude that the data is neither “Overfit” nor “Underfit” in nature. And we can also inference that the model built using grid search CV is best optimized model for prediction. However, we can see significant variations in accuracy score, F1 score, recall values, precision values, ROC curves and AUC scores when compared with default values of KNN and also with model built on balanced dataset. Model built on balance data set using SMOTE technique works well in training dataset however we can see a significant deplete in accuracy when it comes to testing dataset.

BUILDING NAÏVE BAYES MODEL: -

Post splitting data into training and testing we are now ready to build model using Naïve Bayes algorithm. Naïve Bayes algorithm is based out of Bayes theorem of conditional probability. Considering that all events are independent of each other and then we find the probability of an event happening under the condition that one of the events has already occurred.

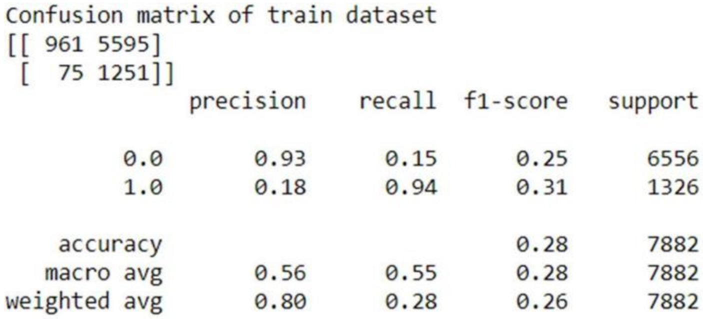
Below are accuracy scores using this model: -



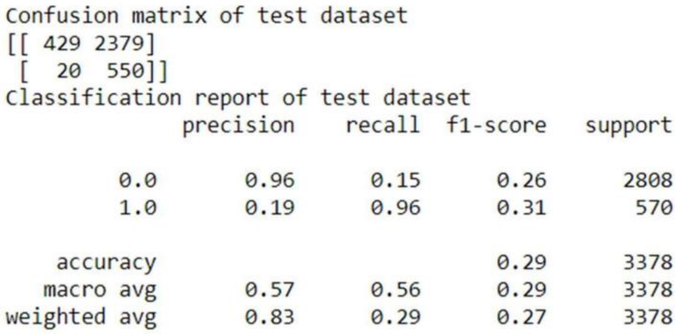
```
Accracy of training dataset: 0.28063943161634103

 Accracy of testing dataset: 0.2898164594434577
```

Fig 47: Training Scores from Naïve Bayes with SMOTE

Below are confusion matrices and classification reports using this model: -



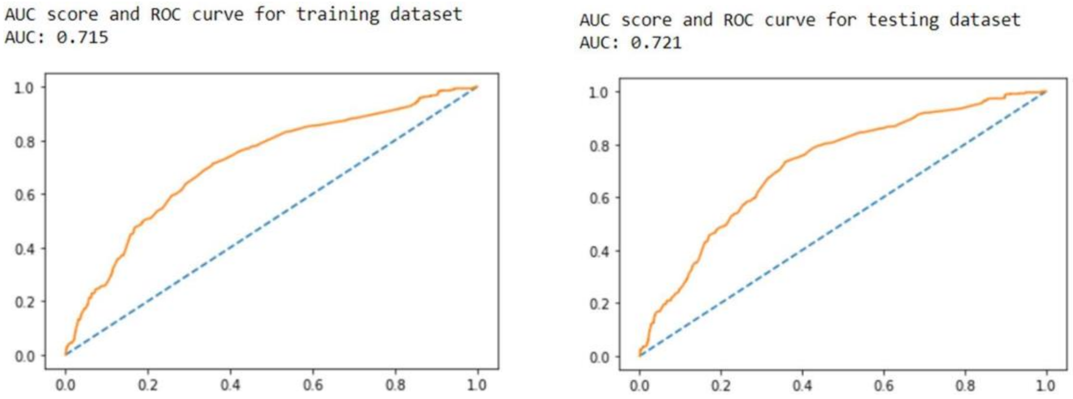
```
Confusion matrix of train dataset
[[ 961 5595]
 [  75 1251]]
             precision    recall  f1-score   support

        0.0       0.93      0.15      0.25      6556
        1.0       0.18      0.94      0.31      1326

   accuracy                           0.28      7882
  macro avg       0.56      0.55      0.28      7882
weighted avg       0.80      0.28      0.26      7882
```



```
Confusion matrix of test dataset
[[ 429 2379]
 [  20  550]]
Classification report of test dataset
              precision    recall  f1-score   support

         0.0       0.96      0.15      0.26      2808
         1.0       0.19      0.96      0.31       570

    accuracy                           0.29      3378
   macro avg       0.57      0.56      0.29      3378
weighted avg       0.83      0.29      0.27      3378
```

Fig 48: Confusion Matrix and Classification Report from Naïve Bayes with SMOTE

Below are AUC scores and ROC curves using this model: -

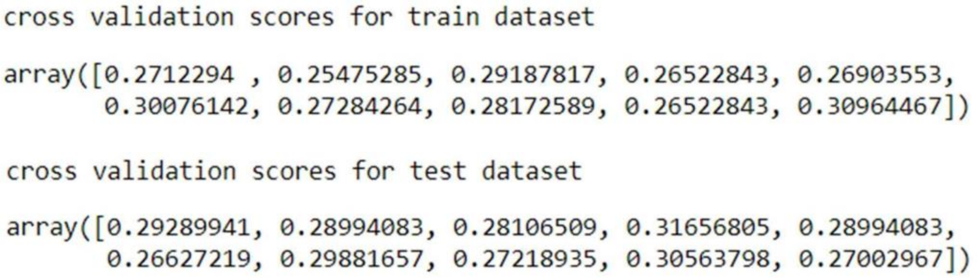


```
AUC score and ROC curve for training dataset
AUC: 0.715
```

```
AUC score and ROC curve for testing dataset
AUC: 0.721
```

Fig 49: ROC Curve and AUC Scores from Naïve Bayes with SMOTE

Below are 10-fold cross validation scores using this model: -



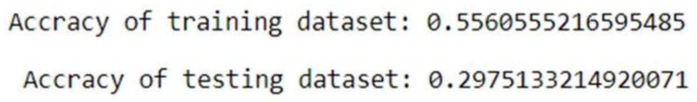
```
cross validation scores for train dataset

array([0.2712294 , 0.25475285, 0.29187817, 0.26522843, 0.26903553,
       0.30076142, 0.27284264, 0.28172589, 0.26522843, 0.30964467])

cross validation scores for test dataset

array([0.29289941, 0.28994083, 0.28106509, 0.31656805, 0.28994083,
       0.26627219, 0.29881657, 0.27218935, 0.30563798, 0.27002967])
```

Fig 50: Cross Validation Scores From Naïve Bayes with SMOTE

Building Gaussian Naive Bayes over balanced data using SMOTE

After building naïve Bayes model using original imbalanced data. Now, we can try and build the same model using balanced data to check if it outperforms in terms of accuracy and other measurement factors.

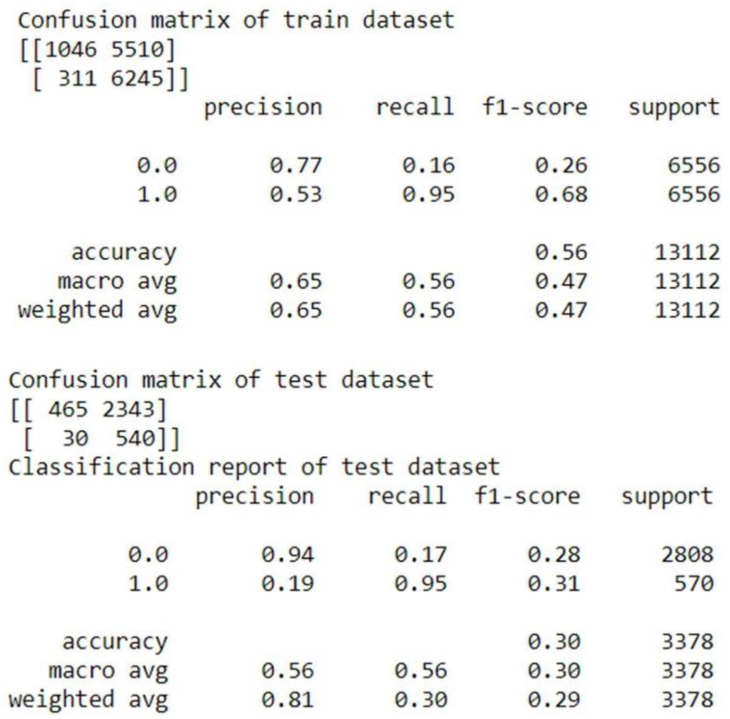
Below are the accuracy scores obtained using Naïve Bayes algorithm over balanced dataset: -



```
Accracy of training dataset: 0.5560555216595485

 Accracy of testing dataset: 0.2975133214920071
```

Fig 51: Accuracy Scores from Naïve Bayes with SMOTE

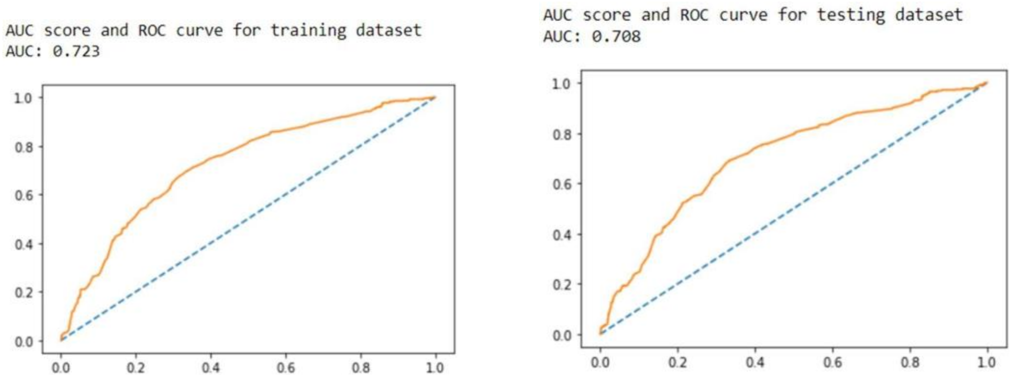
Below are the confusion matrices and classification reports obtained using Naïve Bayes algorithm over balanced dataset: -



```
Confusion matrix of train dataset
[[1046 5510]
 [ 311 6245]]
              precision    recall  f1-score   support

         0.0       0.77      0.16      0.26      6556
         1.0       0.53      0.95      0.68      6556

    accuracy                           0.56     13112
   macro avg       0.65      0.56      0.47     13112
weighted avg       0.65      0.56      0.47     13112

Confusion matrix of test dataset
[[ 465 2343]
 [  30  540]]
Classification report of test dataset
              precision    recall  f1-score   support

         0.0       0.94      0.17      0.28      2808
         1.0       0.19      0.95      0.31       570

    accuracy                           0.30      3378
   macro avg       0.56      0.56      0.30      3378
weighted avg       0.81      0.30      0.29      3378
```

Fig 52: Confusion Matrix and Classification Report From Naïve Bayes with SMOTE

Below are the AUC scores and ROC curves obtained using Naïve Bayes algorithm over balancedataset: -

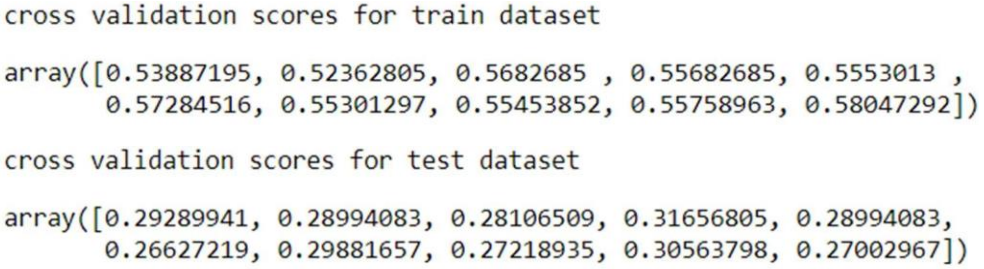


```
AUC score and ROC curve for training dataset
AUC: 0.723
```

```
AUC score and ROC curve for testing dataset
AUC: 0.708
```

Fig 53: ROC Curve and AUC Scores from Naïve Bayes with SMOTE

Below are the 10-fold cross validation scores obtained using Naïve Bayes algorithm over balanced dataset: -



```
cross validation scores for train dataset

array([0.53887195, 0.52362805, 0.5682685 , 0.55682685, 0.5553013 ,
       0.57284516, 0.55301297, 0.55453852, 0.55758963, 0.58047292])

cross validation scores for test dataset

array([0.29289941, 0.28994083, 0.28106509, 0.31656805, 0.28994083,
       0.26627219, 0.29881657, 0.27218935, 0.30563798, 0.27002967])
```

Fig 54: Cross Validation Scores From Naïve Bayes with SMOTE

Inference/Conclusion from Naïve Bayes Model: -

From the above we can conclude that the data is neither “Overfit” nor “Underfit” in nature. And we can also inference that the model built using original imbalanced data is best optimized model for prediction. However, we can see significant variations in accuracy score, F1 score, recall values, precision values, ROC curves and AUC scores when compared with model built on balanced dataset. Model built on balance data set using SMOTE technique works well in training dataset however we can see a significant deplete in accuracy when it comes to training dataset.

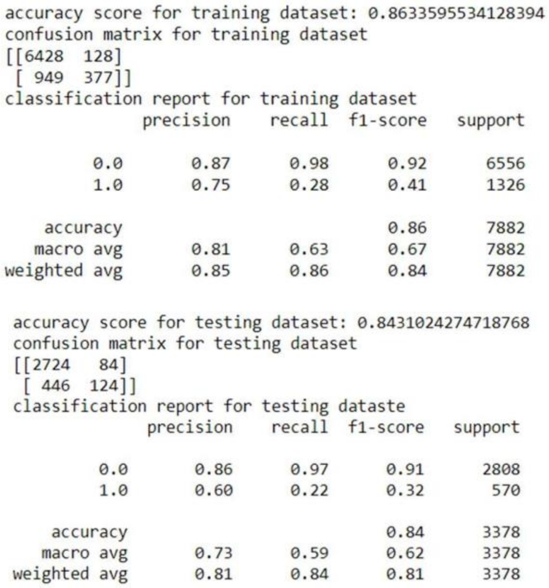
BAGGING: -

Building Random Forest model: -

Firstly, let’s build and random forest model for original dataset and balanced dataset and check the performance for the same.

Random Forest Built in original dataset: -

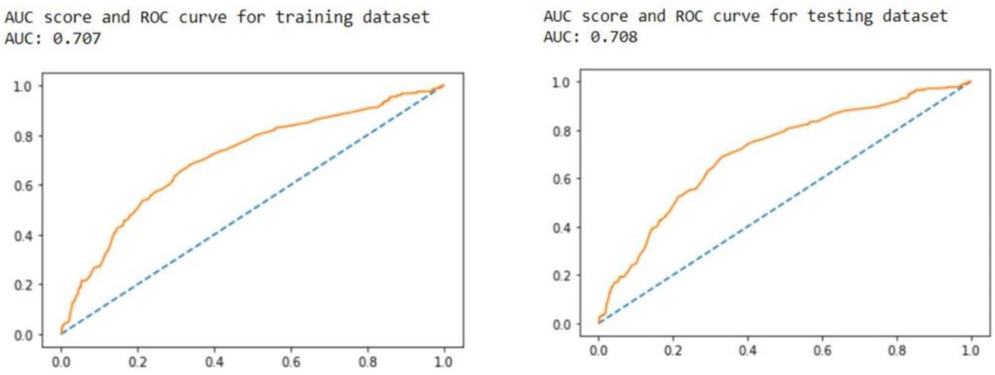
Below are the accuracy scores, confusion matrix and classification report for training and testing dataset: -



```
accuracy score for training dataset: 0.8633595534128394
confusion matrix for training dataset
[[6428  128]
 [ 949  377]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.87      0.98      0.92      6556
         1.0       0.75      0.28      0.41      1326

    accuracy                           0.86      7882
   macro avg       0.81      0.63      0.67      7882
weighted avg       0.85      0.86      0.84      7882

 accuracy score for testing dataset: 0.8431024274718768
 confusion matrix for testing dataset
 [[2724   84]
  [ 446  124]]
 classification report for testing dataste
               precision    recall  f1-score   support

          0.0       0.86      0.97      0.91      2808
          1.0       0.60      0.22      0.32       570

     accuracy                           0.84      3378
    macro avg       0.73      0.59      0.62      3378
 weighted avg       0.81      0.84      0.81      3378
```

Fig 55: Accuracy Parameters from Random Forrest

Below are ROC Curve and AUC Score of train and test data set



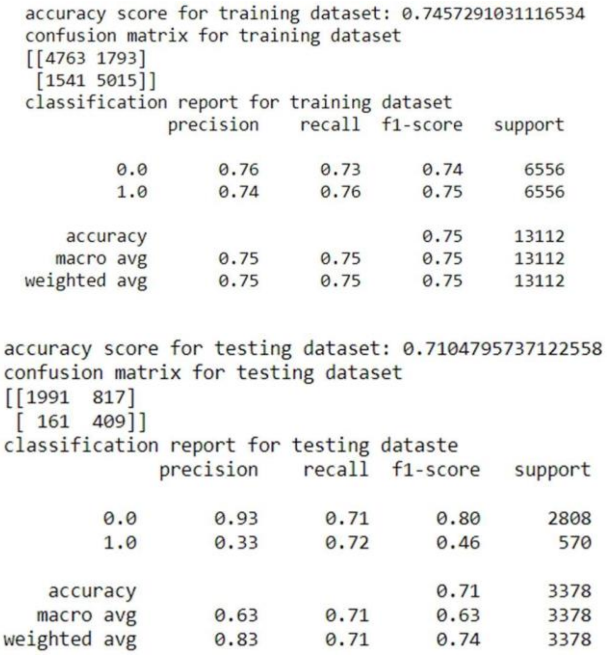
```
AUC score and ROC curve for training dataset
AUC: 0.707
```

```
AUC score and ROC curve for testing dataset
AUC: 0.708
```

Fig 56: ROC Curve and AUC Scores from Random Forrest

Random Forest Built in balanced dataset: -

Below are the accuracy scores, confusion matrix and classification report for training and Tetsing dataset: -



```
accuracy score for training dataset: 0.7457291031116534
confusion matrix for training dataset
[[4763 1793]
 [1541 5015]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.76      0.73      0.74      6556
         1.0       0.74      0.76      0.75      6556

    accuracy                           0.75     13112
   macro avg       0.75      0.75      0.75     13112
weighted avg       0.75      0.75      0.75     13112
```

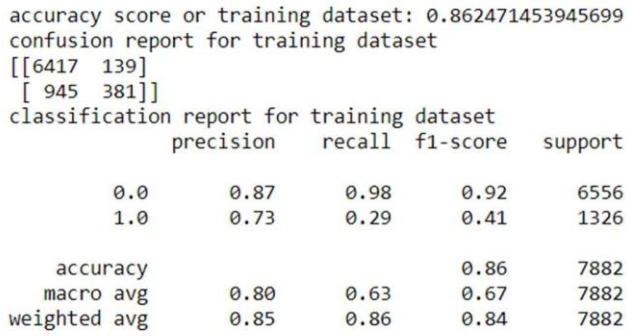
```
accuracy score for testing dataset: 0.7104795737122558
confusion matrix for testing dataset
[[1991  817]
 [ 161  409]]
classification report for testing dataste
              precision    recall  f1-score   support

         0.0       0.93      0.71      0.80      2808
         1.0       0.33      0.72      0.46       570

    accuracy                           0.71      3378
   macro avg       0.63      0.71      0.63      3378
weighted avg       0.83      0.71      0.74      3378
```

Fig 57: Accuracy Parameters from Random Forrest with SMOTE

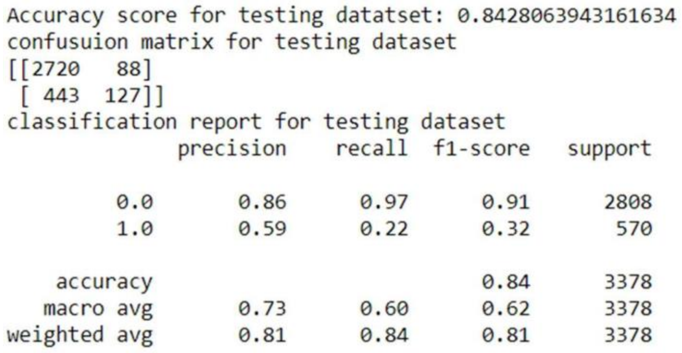
Bagging on original dataset: -

Let’s build bagging model using random forest and check if it can perform better than general random forest model.

Below are the accuracy scores, confusion matrix and classification report for training and Testing dataset: -



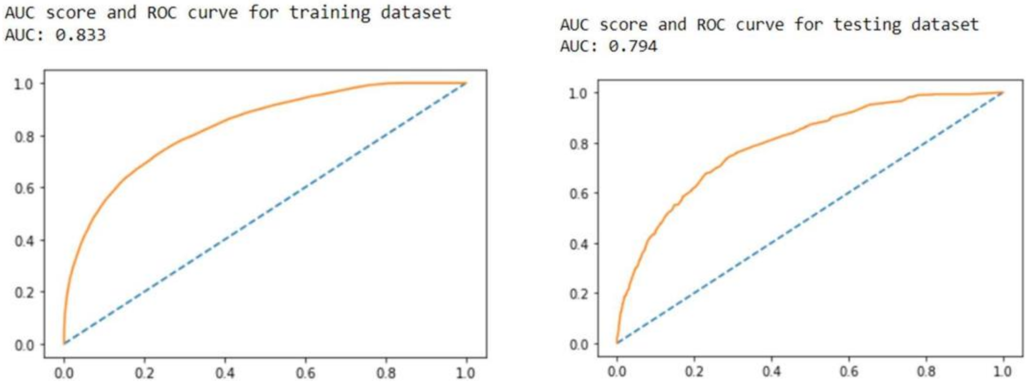
```
accuracy score or training dataset: 0.862471453945699
confusion report for training dataset
[[6417  139]
 [ 945  381]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.87      0.98      0.92      6556
         1.0       0.73      0.29      0.41      1326

    accuracy                           0.86      7882
   macro avg       0.80      0.63      0.67      7882
weighted avg       0.85      0.86      0.84      7882
```



```
Accuracy score for testing datatset: 0.8428063943161634
confusuion matrix for testing dataset
[[2720   88]
 [ 443  127]]
classification report for testing dataset
              precision    recall  f1-score   support

         0.0       0.86      0.97      0.91      2808
         1.0       0.59      0.22      0.32       570

    accuracy                           0.84      3378
   macro avg       0.73      0.60      0.62      3378
weighted avg       0.81      0.84      0.81      3378
```

Fig 58: Bagging On Original Dataset

Below is the AUC score and ROC curve for training and testing dataset: -



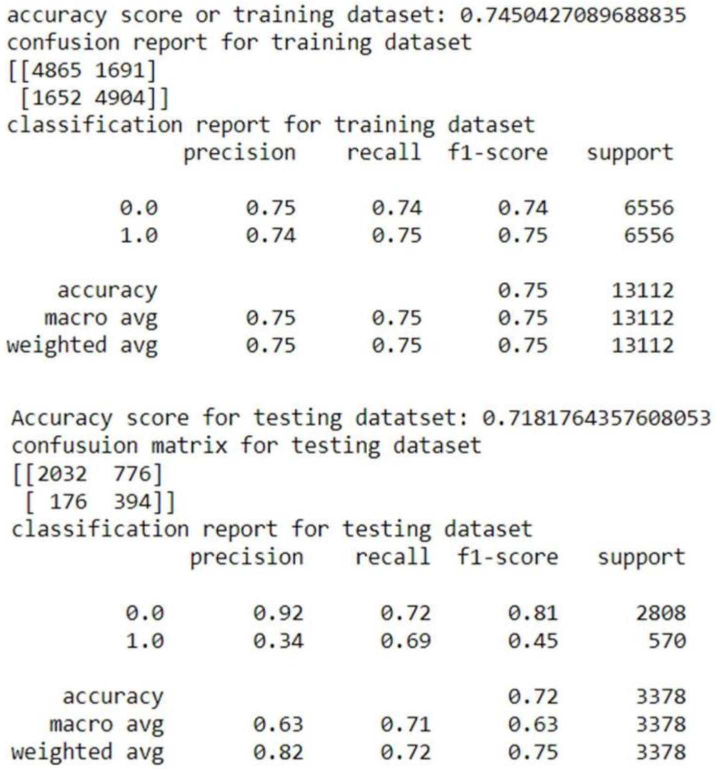
```
AUC score and ROC curve for training dataset
AUC: 0.833
```

```
AUC score and ROC curve for testing dataset
AUC: 0.794
```

Fig 59: ROC Curve and AUC Score Bagging On Original Dataset

Bagging on Balanced dataset: -

Below are the accuracy scores, confusion matrix and classification report for training and Testing dataset:-

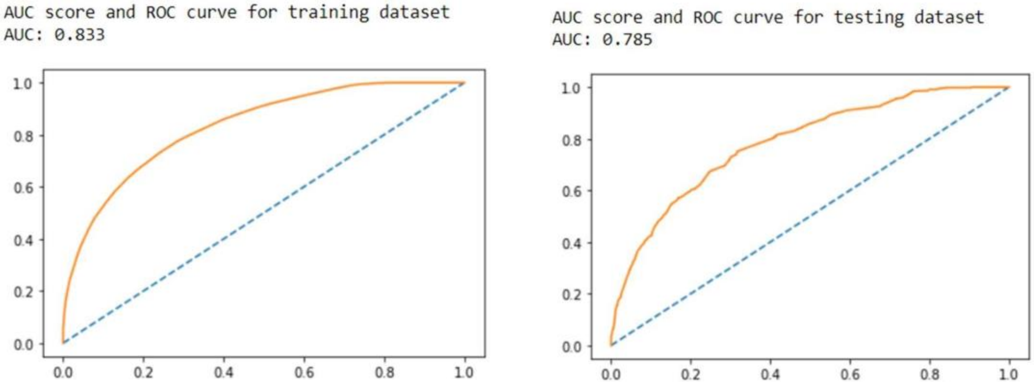


```
accuracy score or training dataset: 0.7450427089688835
confusion report for training dataset
[[4865 1691]
 [1652 4904]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.75      0.74      0.74      6556
         1.0       0.74      0.75      0.75      6556

    accuracy                           0.75     13112
   macro avg       0.75      0.75      0.75     13112
weighted avg       0.75      0.75      0.75     13112
```

```
Accuracy score for testing datatset: 0.7181764357608053
confusuion matrix for testing dataset
[[2032  776]
 [ 176  394]]
classification report for testing dataset
              precision    recall  f1-score   support

         0.0       0.92      0.72      0.81      2808
         1.0       0.34      0.69      0.45       570

    accuracy                           0.72      3378
   macro avg       0.63      0.71      0.63      3378
weighted avg       0.82      0.72      0.75      3378
```

Fig 60: Accuracy Parameters from Bagging on Balanced Dataset

Below is the AUC score and ROC curve for training and testing dataset: -



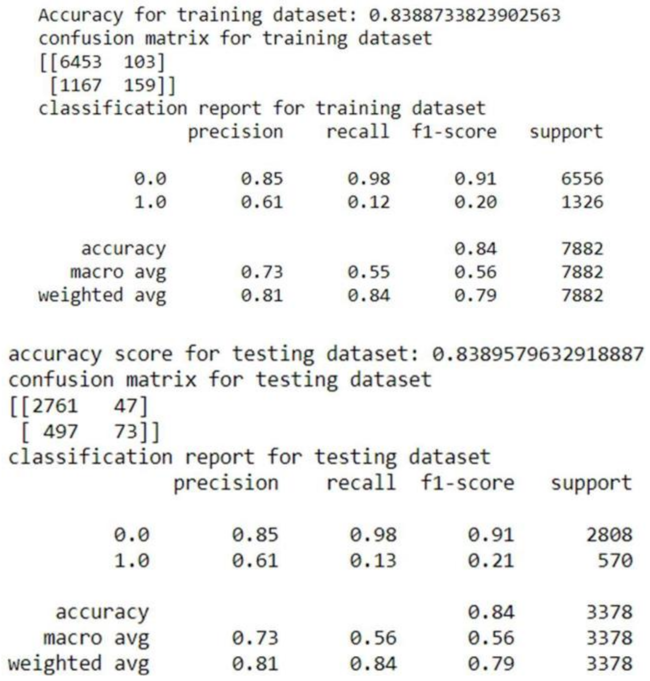
```
AUC score and ROC curve for training dataset
AUC: 0.833
```

```
AUC score and ROC curve for testing dataset
AUC: 0.785
```

Fig 61: ROC Curve and AUC Scores from Bagging on Balanced Dataset

Building Ada-Boost Model on original dataset: -

Below are the accuracy scores, confusion matrix and classification reports for training and Tetsing dataset: -

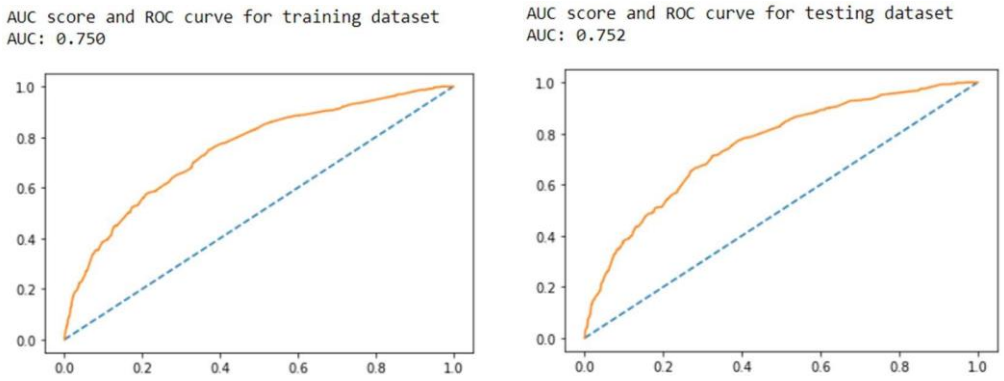


```
Accuracy for training dataset: 0.8388733823902563
confusion matrix for training dataset
[[6453  103]
 [1167  159]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.85      0.98      0.91      6556
         1.0       0.61      0.12      0.20      1326

    accuracy                           0.84      7882
   macro avg       0.73      0.55      0.56      7882
weighted avg       0.81      0.84      0.79      7882
```

```
accuracy score for testing dataset: 0.8389579632918887
confusion matrix for testing dataset
[[2761   47]
 [ 497   73]]
classification report for testing dataset
              precision    recall  f1-score   support

         0.0       0.85      0.98      0.91      2808
         1.0       0.61      0.13      0.21       570

    accuracy                           0.84      3378
   macro avg       0.73      0.56      0.56      3378
weighted avg       0.81      0.84      0.79      3378
```

Fig 62: Accuracy Parameters from Ada-Boosting on Original Dataset

Below are the AUC scores and ROC curve for training and testing dataset: -



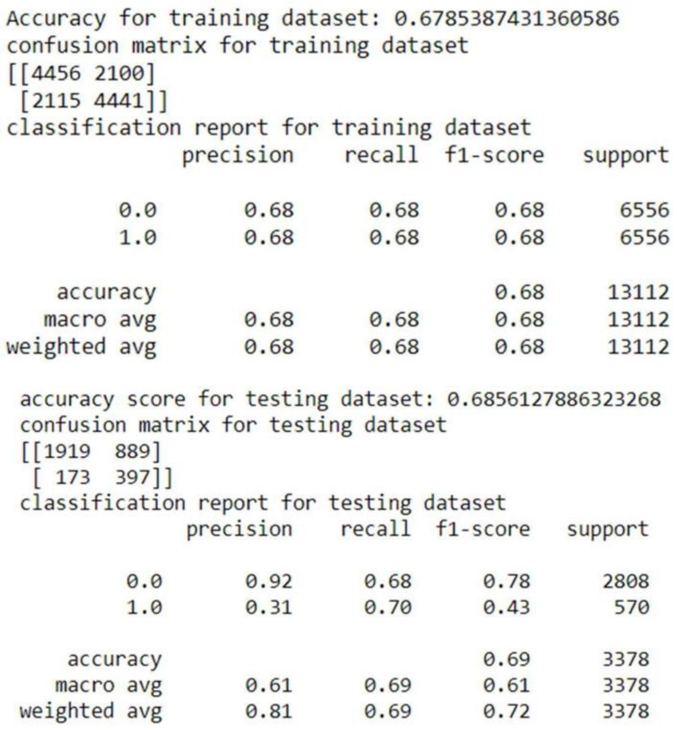
```
AUC score and ROC curve for training dataset
AUC: 0.750
```

```
AUC score and ROC curve for testing dataset
AUC: 0.752
```

Fig 63: ROC curve and AUC score from Ada-Boosting on Original Dataset

Building Ada-Boost Model on balanced dataset: -

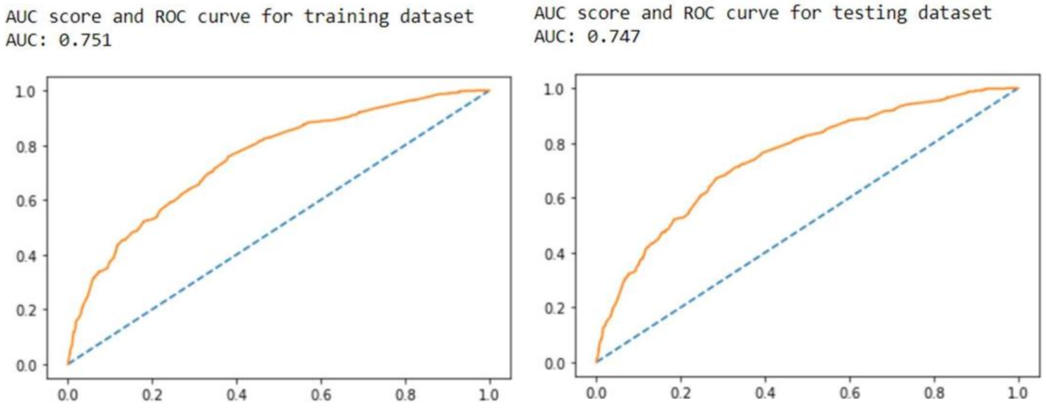
Below are the accuracy scores, confusion matrix and classification reports for training and Testing dataset: -



```
Accuracy for training dataset: 0.6785387431360586
confusion matrix for training dataset
[[4456 2100]
 [2115 4441]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.68      0.68      0.68      6556
         1.0       0.68      0.68      0.68      6556

    accuracy                           0.68     13112
   macro avg       0.68      0.68      0.68     13112
weighted avg       0.68      0.68      0.68     13112

 accuracy score for testing dataset: 0.6856127886323268
 confusion matrix for testing dataset
 [[1919  889]
  [ 173  397]]
 classification report for testing dataset
               precision    recall  f1-score   support

          0.0       0.92      0.68      0.78      2808
          1.0       0.31      0.70      0.43       570

     accuracy                           0.69      3378
    macro avg       0.61      0.69      0.61      3378
 weighted avg       0.81      0.69      0.72      3378
```

Fig 64: Accuracy Parameter from Ada-Boosting on Balanced Dataset

Below are the AUC scores and ROC curve for training and testing dataset: -



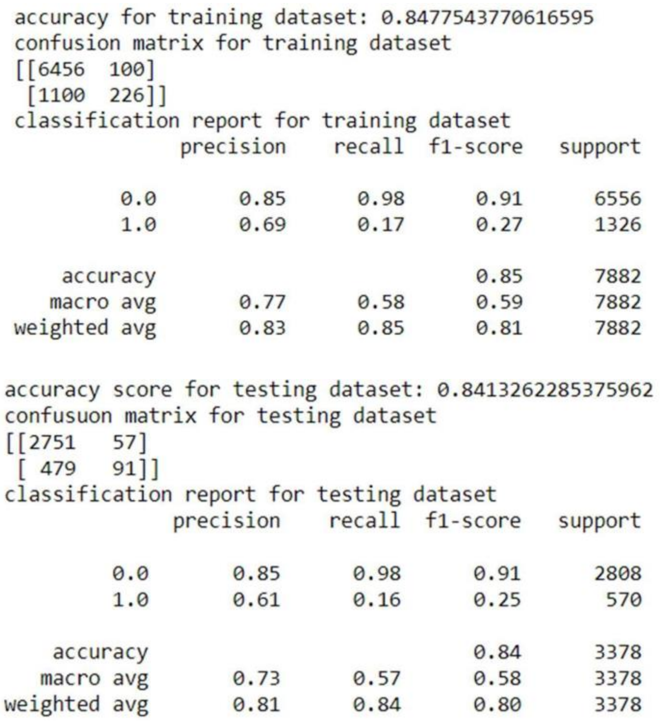
```
AUC score and ROC curve for training dataset
AUC: 0.751
```

```
AUC score and ROC curve for testing dataset
AUC: 0.747
```

Fig 65: ROC Curve and AUC Score from Ada-Boosting on Balanced Dataset

<u>Building Gradient Boosting Model on original dataset: -</u>

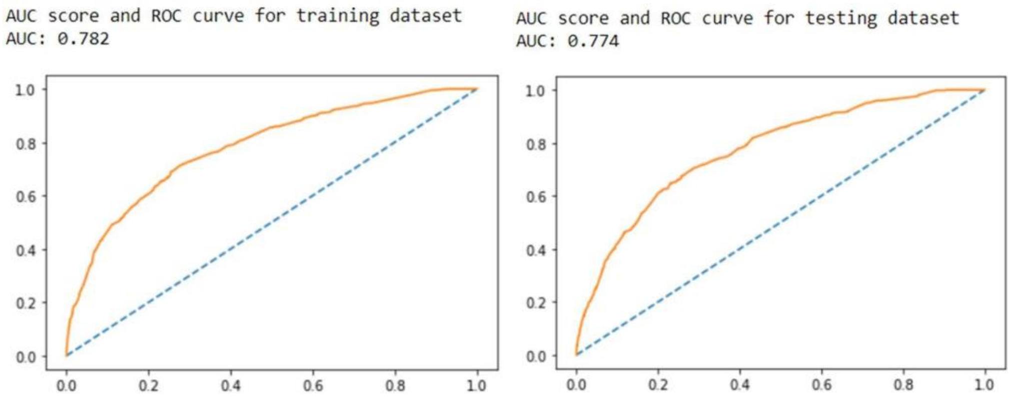
<u>Below are the accuracy scores, confusion matrix and classification reports for training andTesting dataset: -</u>



```
accuracy for training dataset: 0.8477543770616595
confusion matrix for training dataset
[[6456  100]
 [1100  226]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.85      0.98      0.91      6556
         1.0       0.69      0.17      0.27      1326

    accuracy                           0.85      7882
   macro avg       0.77      0.58      0.59      7882
weighted avg       0.83      0.85      0.81      7882

accuracy score for testing dataset: 0.8413262285375962
confusuon matrix for testing dataset
[[2751   57]
 [ 479   91]]
classification report for testing dataset
              precision    recall  f1-score   support

         0.0       0.85      0.98      0.91      2808
         1.0       0.61      0.16      0.25       570

    accuracy                           0.84      3378
   macro avg       0.73      0.57      0.58      3378
weighted avg       0.81      0.84      0.80      3378
```

Fig 66: Accuracy Parameter from Gradient-Boosting on Original Dataset

<u>Below are the AUC scores and ROC curve for training and testing dataset: -</u>



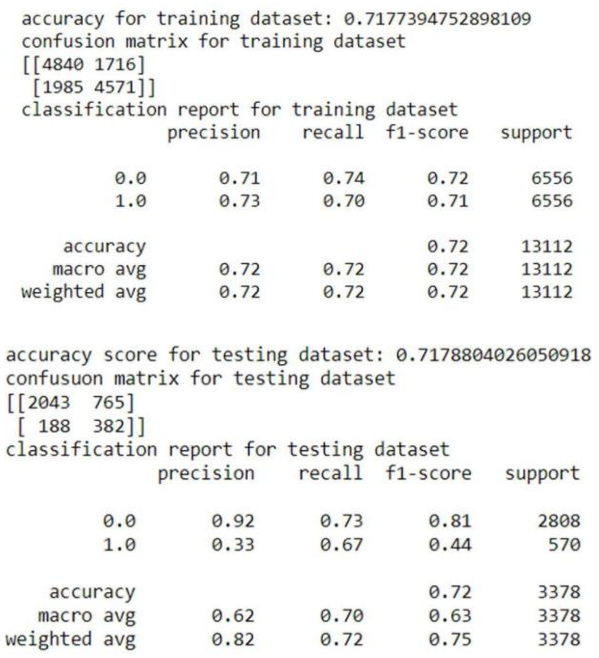
```
AUC score and ROC curve for training dataset
AUC: 0.782
```

```
AUC score and ROC curve for testing dataset
AUC: 0.774
```

Fig 67: ROC Curve and AUC Score from Gradient-Boosting on Original Dataset

Building Gradient Boosting Model on balanced dataset: -

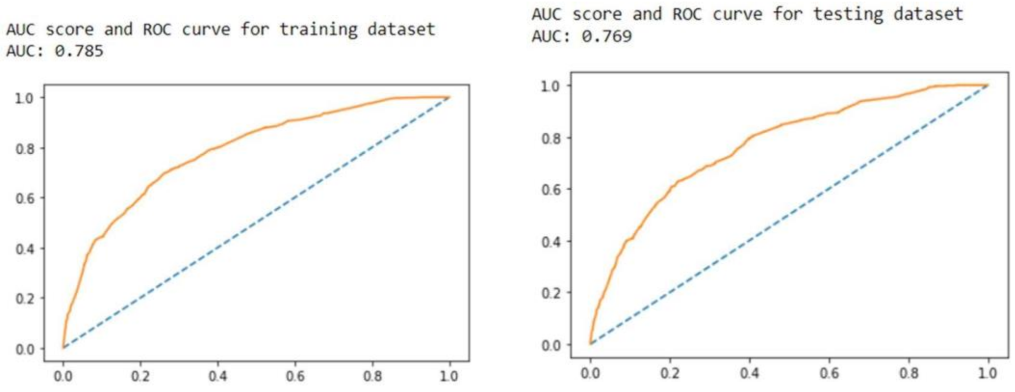
Below are the accuracy scores, confusion matrix and classification reports for training and Testing dataset: -



```
accuracy for training dataset: 0.7177394752898109
confusion matrix for training dataset
[[4840 1716]
 [1985 4571]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.71      0.74      0.72      6556
         1.0       0.73      0.70      0.71      6556

    accuracy                           0.72     13112
   macro avg       0.72      0.72      0.72     13112
weighted avg       0.72      0.72      0.72     13112

accuracy score for testing dataset: 0.7178804026050918
confusuon matrix for testing dataset
[[2043  765]
 [ 188  382]]
classification report for testing dataset
              precision    recall  f1-score   support

         0.0       0.92      0.73      0.81      2808
         1.0       0.33      0.67      0.44       570

    accuracy                           0.72      3378
   macro avg       0.62      0.70      0.63      3378
weighted avg       0.82      0.72      0.75      3378
```

Fig 68: Accuracy Parameter from Gradient-Boosting on Balanced Dataset

Below are the AUC scores and ROC curve for training and testing dataset: -



AUC score and ROC curve for training dataset
AUC: 0.785

AUC score and ROC curve for testing dataset
AUC: 0.769

Fig 69: ROC Curve and AUC Score from Gradient-Boosting on Balanced Dataset

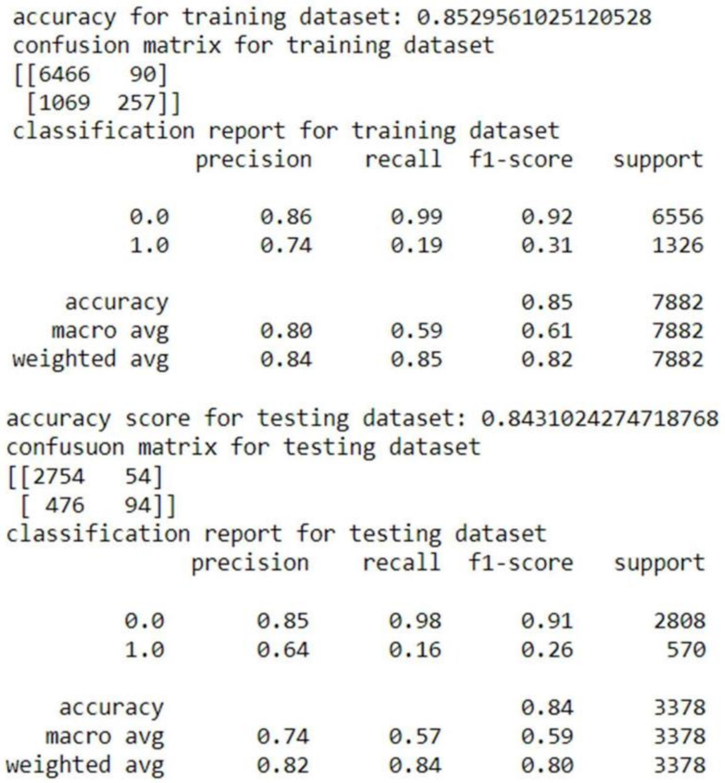
Inferences from Bagging and Boosting model: -

- Form both the model above we can notice that the model is still outfitted.
- When to come to model performance over balanced dataset it performs well in training

dataset however the accuracy reduces when it comes to training dataset.

Building Support Vector Machine (SVM) Model on Original dataset: -

Below are the accuracy scores, confusion matrix and classification reports for training and Testing dataset: -

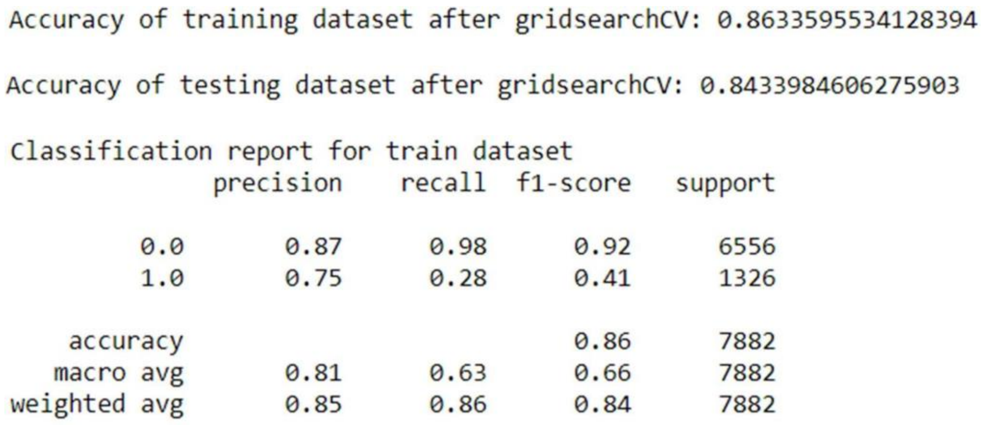


```
accuracy for training dataset: 0.8529561025120528
confusion matrix for training dataset
[[6466   90]
 [1069  257]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.86      0.99      0.92      6556
         1.0       0.74      0.19      0.31      1326

    accuracy                           0.85      7882
   macro avg       0.80      0.59      0.61      7882
weighted avg       0.84      0.85      0.82      7882

accuracy score for testing dataset: 0.8431024274718768
confusuon matrix for testing dataset
[[2754   54]
 [ 476   94]]
classification report for testing dataset
              precision    recall  f1-score   support

         0.0       0.85      0.98      0.91      2808
         1.0       0.64      0.16      0.26       570

    accuracy                           0.84      3378
   macro avg       0.74      0.57      0.59      3378
weighted avg       0.82      0.84      0.80      3378
```

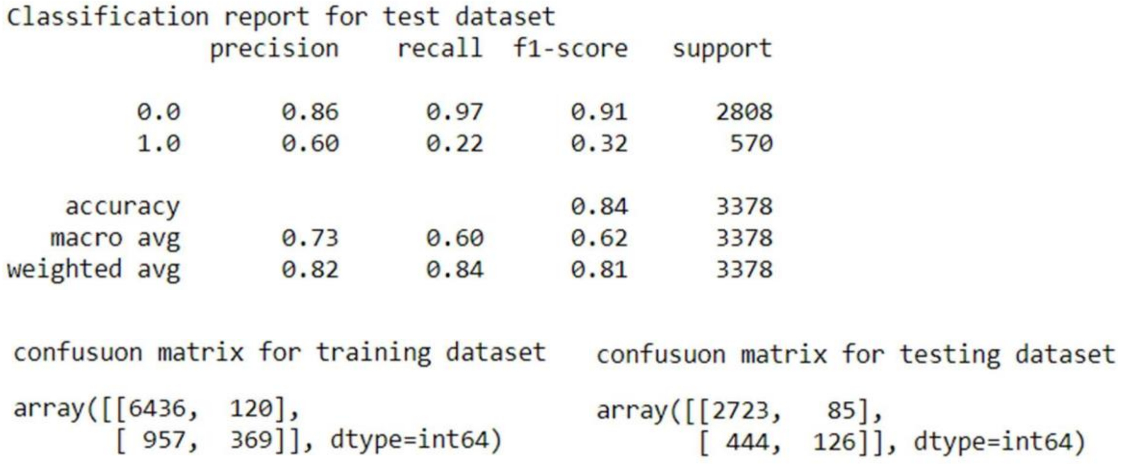
Fig 70: Accuracy parameter Scores from SVM on Original Dataset

Building Support Vector Machine (SVM) Model on Original dataset Using Hyper-Parameter: -

Below are the accuracy scores, confusion matrix and classification reports for training and Testing dataset: -



```
Accuracy of training dataset after gridsearchCV: 0.8633595534128394

Accuracy of testing dataset after gridsearchCV: 0.8433984606275903

Classification report for train dataset
              precision    recall  f1-score   support

         0.0       0.87      0.98      0.92      6556
         1.0       0.75      0.28      0.41      1326

    accuracy                           0.86      7882
   macro avg       0.81      0.63      0.66      7882
weighted avg       0.85      0.86      0.84      7882
```



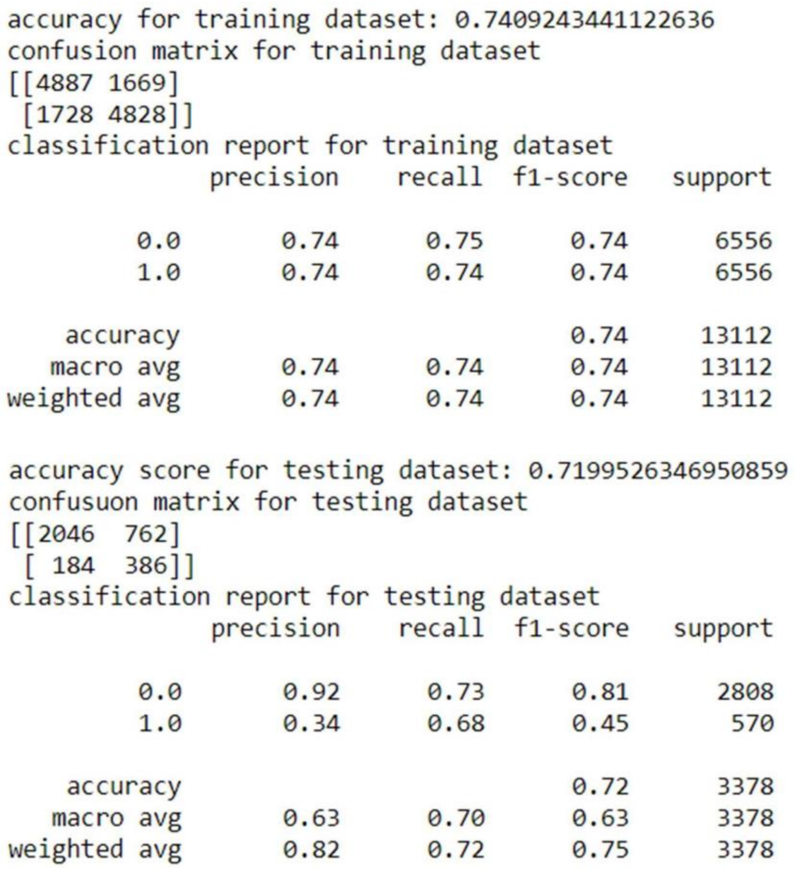
```
Classification report for test dataset
              precision    recall  f1-score   support

         0.0       0.86      0.97      0.91      2808
         1.0       0.60      0.22      0.32       570

    accuracy                           0.84      3378
   macro avg       0.73      0.60      0.62      3378
weighted avg       0.82      0.84      0.81      3378
```

```
confusuon matrix for training dataset    confusuon matrix for testing dataset

array([[6436,  120],                     array([[2723,   85],
       [ 957,  369]], dtype=int64)              [ 444,  126]], dtype=int64)
```

Fig 71: Accuracy parameter Scores from SVM on Original Dataset Using Hyper-Parameter

Building Support Vector Machine (SVM) Model on Balanced Dataset: -

Below are the accuracy scores, confusion matrix and classification reports for training and Testing dataset: -



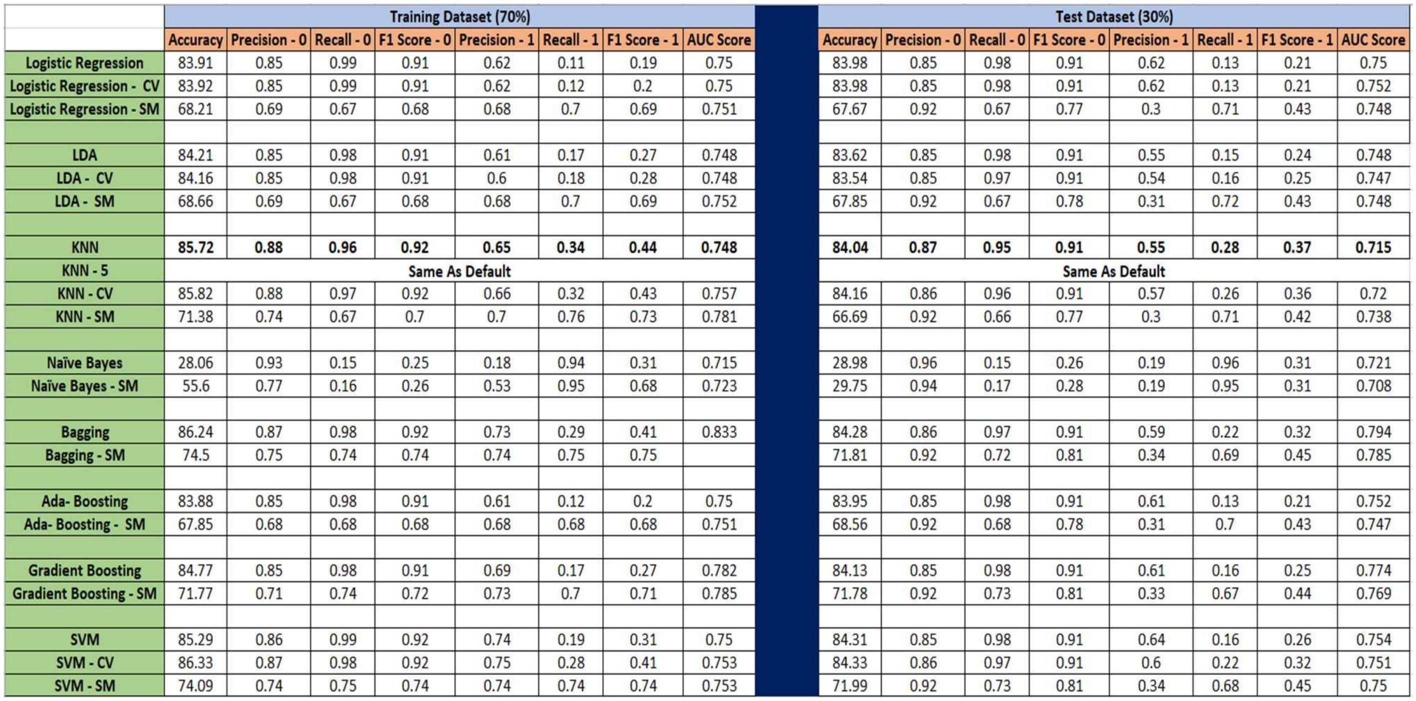
```
accuracy for training dataset: 0.7409243441122636
confusion matrix for training dataset
[[4887 1669]
 [1728 4828]]
classification report for training dataset
              precision    recall  f1-score   support

         0.0       0.74      0.75      0.74      6556
         1.0       0.74      0.74      0.74      6556

    accuracy                           0.74     13112
   macro avg       0.74      0.74      0.74     13112
weighted avg       0.74      0.74      0.74     13112

accuracy score for testing dataset: 0.7199526346950859
confusuon matrix for testing dataset
[[2046  762]
 [ 184  386]]
classification report for testing dataset
              precision    recall  f1-score   support

         0.0       0.92      0.73      0.81      2808
         1.0       0.34      0.68      0.45       570

    accuracy                           0.72      3378
   macro avg       0.63      0.70      0.63      3378
weighted avg       0.82      0.72      0.75      3378
```

Fig 72: Accuracy parameter Scores from SVM on Balanced Dataset

Inferences from SVM model: -

From the above we can conclude that the data is neither “Overfit” nor “Underfit” in nature. And we can also inference that the model built using balanced data is best optimized modelfor prediction. However, we can see significant variations in accuracy score, F1 score, recall values, precision values, ROC curves and AUC scores when compared with model built on imbalanced dataset. Model built on balance data set using SMOTE technique works well in training dataset and testing dataset.

Overall Model Building Comparison across parameters: -



	Training Dataset (70%)									Test Dataset (30%)							
	Accuracy	Precision - 0	Recall - 0	F1 Score - 0	Precision - 1	Recall - 1	F1 Score - 1	AUC Score		Accuracy	Precision - 0	Recall - 0	F1 Score - 0	Precision - 1	Recall - 1	F1 Score - 1	AUC Score
Logistic Regression	83.91	0.85	0.99	0.91	0.62	0.11	0.19	0.75		83.98	0.85	0.98	0.91	0.62	0.13	0.21	0.75
Logistic Regression - CV	83.92	0.85	0.99	0.91	0.62	0.12	0.2	0.75		83.98	0.85	0.98	0.91	0.62	0.13	0.21	0.752
Logistic Regression - SM	68.21	0.69	0.67	0.68	0.68	0.7	0.69	0.751		67.67	0.92	0.67	0.77	0.3	0.71	0.43	0.748
LDA	84.21	0.85	0.98	0.91	0.61	0.17	0.27	0.748		83.62	0.85	0.98	0.91	0.55	0.15	0.24	0.748
LDA - CV	84.16	0.85	0.98	0.91	0.6	0.18	0.28	0.748		83.54	0.85	0.97	0.91	0.54	0.16	0.25	0.747
LDA - SM	68.66	0.69	0.67	0.68	0.68	0.7	0.69	0.752		67.85	0.92	0.67	0.78	0.31	0.72	0.43	0.748
KNN	**85.72**	**0.88**	**0.96**	**0.92**	**0.65**	**0.34**	**0.44**	**0.748**		**84.04**	**0.87**	**0.95**	**0.91**	**0.55**	**0.28**	**0.37**	**0.715**
KNN - 5	Same As Default									Same As Default							
KNN - CV	85.82	0.88	0.97	0.92	0.66	0.32	0.43	0.757		84.16	0.86	0.96	0.91	0.57	0.26	0.36	0.72
KNN - SM	71.38	0.74	0.67	0.7	0.7	0.76	0.73	0.781		66.69	0.92	0.66	0.77	0.3	0.71	0.42	0.738
Naïve Bayes	28.06	0.93	0.15	0.25	0.18	0.94	0.31	0.715		28.98	0.96	0.15	0.26	0.19	0.96	0.31	0.721
Naïve Bayes - SM	55.6	0.77	0.16	0.26	0.53	0.95	0.68	0.723		29.75	0.94	0.17	0.28	0.19	0.95	0.31	0.708
Bagging	86.24	0.87	0.98	0.92	0.73	0.29	0.41	0.833		84.28	0.86	0.97	0.91	0.59	0.22	0.32	0.794
Bagging - SM	74.5	0.75	0.74	0.74	0.74	0.75	0.75			71.81	0.92	0.72	0.81	0.34	0.69	0.45	0.785
Ada- Boosting	83.88	0.85	0.98	0.91	0.61	0.12	0.2	0.75		83.95	0.85	0.98	0.91	0.61	0.13	0.21	0.752
Ada- Boosting - SM	67.85	0.68	0.68	0.68	0.68	0.68	0.68	0.751		68.56	0.92	0.68	0.78	0.31	0.7	0.43	0.747
Gradient Boosting	84.77	0.85	0.98	0.91	0.69	0.17	0.27	0.782		84.13	0.85	0.98	0.91	0.61	0.16	0.25	0.774
Gradient Boosting - SM	71.77	0.71	0.74	0.72	0.73	0.7	0.71	0.785		71.78	0.92	0.73	0.81	0.33	0.67	0.44	0.769
SVM	85.29	0.86	0.99	0.92	0.74	0.19	0.31	0.75		84.31	0.85	0.98	0.91	0.64	0.16	0.26	0.754
SVM - CV	86.33	0.87	0.98	0.92	0.75	0.28	0.41	0.753		84.33	0.86	0.97	0.91	0.6	0.22	0.32	0.751
SVM - SM	74.09	0.74	0.75	0.74	0.74	0.74	0.74	0.753		71.99	0.92	0.73	0.81	0.34	0.68	0.45	0.75

Table 9: Comparison across Various Models

Indicators/symbols for above tabular data: -

- CV: - indicates scores for model built on best params obtained from GridSearchCV with model name as prefix.
- SM: - indicates scores for model built on balanced dataset with model name as prefix.
- KNN-5: - Indicates KNN model built with N_neighbors as “5”.

Inferences on final model: -

- From the above tabular representation of all the scores for training and testing dataset across various model we can conclude that the KNN model with default values of hyper-parameters is best optimized for the given dataset. (highlighted in BOLD)
- There is marginal difference in accuracy for Logistic regression and LDA, but comparatively LDA had a little better performance than logistic regression.
- Model with bagging and boosting is also well optimized but difference in accuracy fortraining and testing dataset is little on the higher side as compared to KNN.

- Other models namely Naïve Bayes, LDA and SVM worked well on training dataset but the accuracy came down when performed over testing dataset. Which indicates overfitting of data in that model.
- All models built on balances dataset showed overfitting.
- **We also understand that the accuracy and other measuring parameter of a model can be improved by trying various other combinations of hyper-parameter. Modelbuilding is an iterative process. Model performance both on training and testing dataset can be improves**

Implication of final model on Business: -

- Using the model built above business can plan various strategies to make customers stick with them.
- They can roll out different Offers and discounts as family floater.
- They can give regular discount coupons if paid by their e-wallet platform.
- Discount vouchers of other vendors or on next bill can be provided based in minimum bill criteria.
- This model gives business an idea where they stand currently and what best they can do to improve on the same.