Capstone Project Notes - 2
DSBA

Contents-

List of Figure-

1. Model Building-

In this business case, the need is to predict whether a given customer would churn or not. 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.

1.2 Methodology-

- Different treatments of pre-processed data were prepared - with and without outliers, with and without SMOTE resampling, with and without scaling.
- VIF (Variance Inflation Factor) was calculated for all the predictor variables. One by one predictor variables whose VIF was more than 5 were identified. 4 variables – Cashback, Service Score, Clusters and User count had VIF greater than 5.
- It is to be noted that rev_growth_yoy and coupon_used_for_payment were dropped after Anova and Chi-square test were conducted and double checked against EDA plots. Hence none of the models have used these variables as predictors.
- Scaled data was used for distance-based algorithms such as KNN.
- Data was split into train and test sets in the ratio 70:30. 7882 records were assigned to the training dataset and 3378 records were assigned to the test dataset.

2. Building Logistic Regression Model -

2.1 Building Model with Default hyperparameters-

Post splitting data into a training and testing data set we fitted a logistic regression model into the 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 lbfgs.

Below are the accuracy scores obtained from this model: -

```
Accuracy of training dataset: 0.8920324790662268
```

```
Accuracy of testing dataset: 0.8913558318531676
```

Below is the confusion matrix obtain from the model-

```
Confusion Matrix for train dataset
array([[6358,  197],
       [ 654,  673]])
```

```
Confusion Matrix for test dataset
array([[2729,   80],
       [ 287,  282]])
```

Below is the classification report obtain from this model-

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

           0       0.91      0.97      0.94      6555
           1       0.77      0.51      0.61      1327

    accuracy                           0.89      7882
   macro avg       0.84      0.74      0.77      7882
weighted avg       0.88      0.89      0.88      7882
```

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

           0       0.90      0.97      0.94      2809
           1       0.78      0.50      0.61       569

    accuracy                           0.89      3378
   macro avg       0.84      0.73      0.77      3378
weighted avg       0.88      0.89      0.88      3378
```

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

```
AUC score and ROC curve for training dataset-
AUC: 0.884
AUC score and ROC curve for testing dataset-
AUC: 0.884
```

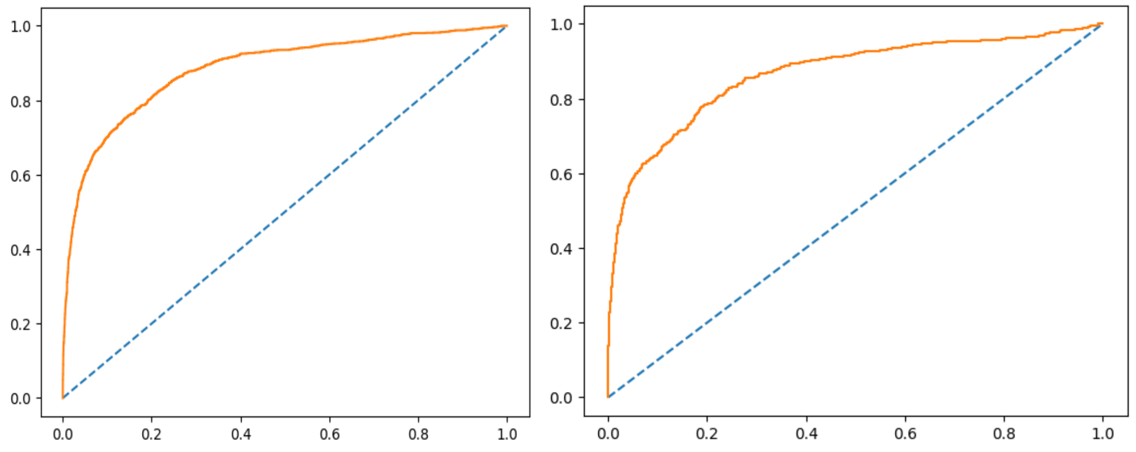


Fig.1.1 ROC curve and AUC score from Logistic Regression

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 Scores for training dataset-

```
array([0.878327  , 0.90240811, 0.88832487, 0.8857868 , 0.88451777,
       0.88705584, 0.87436548, 0.89974619, 0.89467005, 0.90482234])
```

Cross-Validation Scores for testing dataset-

```
array([0.92011834, 0.84615385, 0.87573964, 0.90828402, 0.89349112,
       0.87573964, 0.88757396, 0.8964497 , 0.88724036, 0.89317507])
```

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

2.2 Building Model using GridSearchCV and analysing the best parameters-

We use GridSearchCV to find an optimal combination of hyperparameters that minimizes a predefined loss function to give better results. Performed GridSearchCV with various hyperparameters like “solver”, “penalty” and “tol” and we can find that solver along with “none” penalty worked as best parameters for this dataset.

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

```
Accuracy of training dataset after gridsearchCV: 0.8903831514843948
```

```
Accuracy of testing dataset after gridsearchCV: 0.8904677323860273
```

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

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

           0       0.91      0.97      0.94      6555
           1       0.77      0.50      0.61      1327

    accuracy                           0.89      7882
   macro avg       0.84      0.73      0.77      7882
weighted avg       0.88      0.89      0.88      7882
```

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

           0       0.90      0.97      0.94      2809
           1       0.78      0.49      0.60       569

    accuracy                           0.89      3378
   macro avg       0.84      0.73      0.77      3378
weighted avg       0.88      0.89      0.88      3378
```

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

```
confusion matrix for train dataset
array([[6356,  199],
       [ 665,  662]])
```

```
confusion matrix for test dataset
array([[2728,   81],
       [ 289,  280]])
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.884
AUC score and ROC curve for testing dataset
AUC: 0.867
```

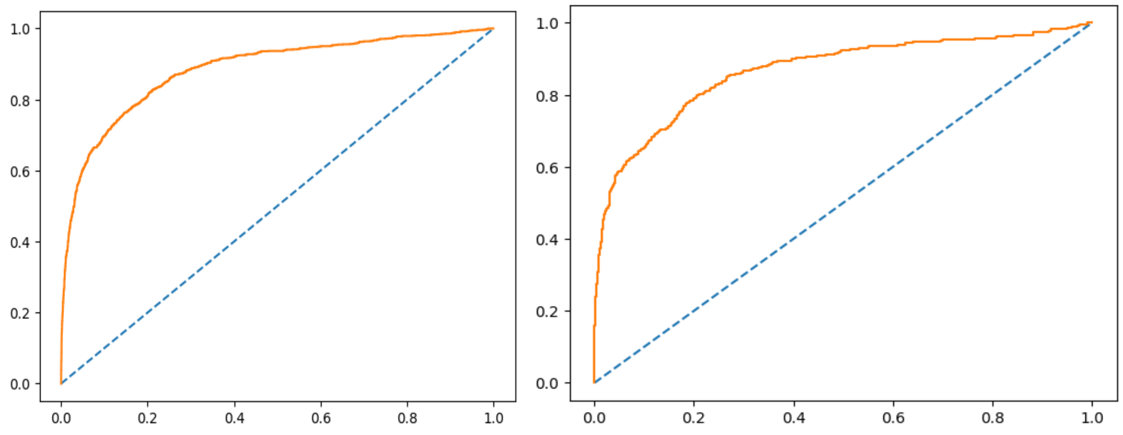


Fig.1.2 ROC Curve and AUC Score From Logistic Regression Using hyper-parameter

Below are the 10-fold cross validation scores-

```
cross validation score for training dataset
array([0.87959442, 0.89987326, 0.88705584, 0.89213198, 0.8857868 ,
       0.88832487, 0.87436548, 0.89974619, 0.89593909, 0.90482234])

cross calidation score for testing dataset
array([0.91420118, 0.84911243, 0.87573964, 0.90828402, 0.8964497 ,
       0.87573964, 0.88757396, 0.89940828, 0.884273  , 0.90207715])
```

2.3 Building Logistic regression model using SMOTE-

In our previous analysis we have seen that the data is imbalanced in nature. We can use the SMOTE technique to balance the data and then tried building a model on the balanced data to check if we can see some significant improvement in accuracy for training and testing the dataset. After building a model on a 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: 0.8093058733790999

Accuracy of testing dataset: 0.7948490230905861
```

Below is the confusion matrix obtained from balanced data-

```
Confusion matrix for train dataset    Confusion matrix for test dataset
array([[5237, 1318],                  array([[2236,  573],
       [1182, 5373]])                        [ 120,  449]])
```

Below is the classification report obtained from balanced data-

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

           0       0.82      0.80      0.81      6555
           1       0.80      0.82      0.81      6555

    accuracy                           0.81     13110
   macro avg       0.81      0.81      0.81     13110
weighted avg       0.81      0.81      0.81     13110


Classification report for test dataset
              precision    recall  f1-score   support

           0       0.95      0.80      0.87      2809
           1       0.44      0.79      0.56       569

    accuracy                           0.79      3378
   macro avg       0.69      0.79      0.72      3378
weighted avg       0.86      0.79      0.82      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.889
AUC score and ROC curve for testing dataset
AUC: 0.866
```

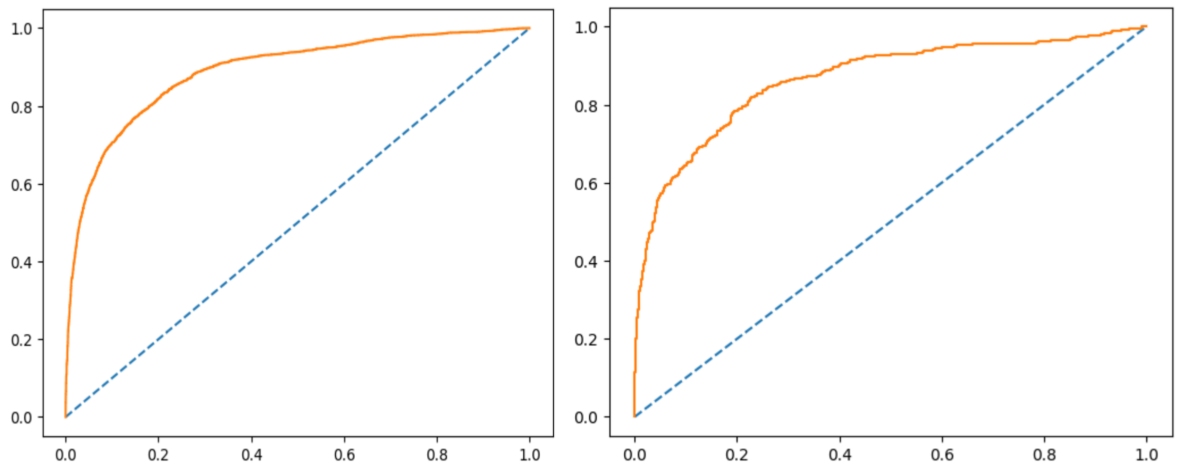


Fig 1.3 ROC Curve & AUC Scores From Logistic Regression with SMOTE

Below are the 10-fold cross validation scores-

Cross validation score for balanced training dataset-

```
array([0.80396644, 0.77955759, 0.80625477, 0.8085431 , 0.80396644,
       0.81922197, 0.81998474, 0.79328757, 0.82227307, 0.8215103 ])
```

Cross validation score for balanced test dataset-

```
array([0.92011834, 0.84615385, 0.87573964, 0.90828402, 0.89349112,
       0.87573964, 0.88757396, 0.8964497 , 0.88724036, 0.89317507])
```

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

Inference 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 a balanced dataset using SMOTE technique works well in training dataset however we can see a significant deplete in accuracy when it comes to testing dataset.

3. Building Linear Discriminant Analysis Model (LDA)-

3.1 Building model with default hyperparameters-

From the above split into training and testing dataset we are building Linear Discriminant Analysis (LDA) model to check if this can outperform Logistic regression model and we can choose form best for further predictions. Firstly, we are building a model using default values of LDA.

Below are the accuracy scores obtained from this model-

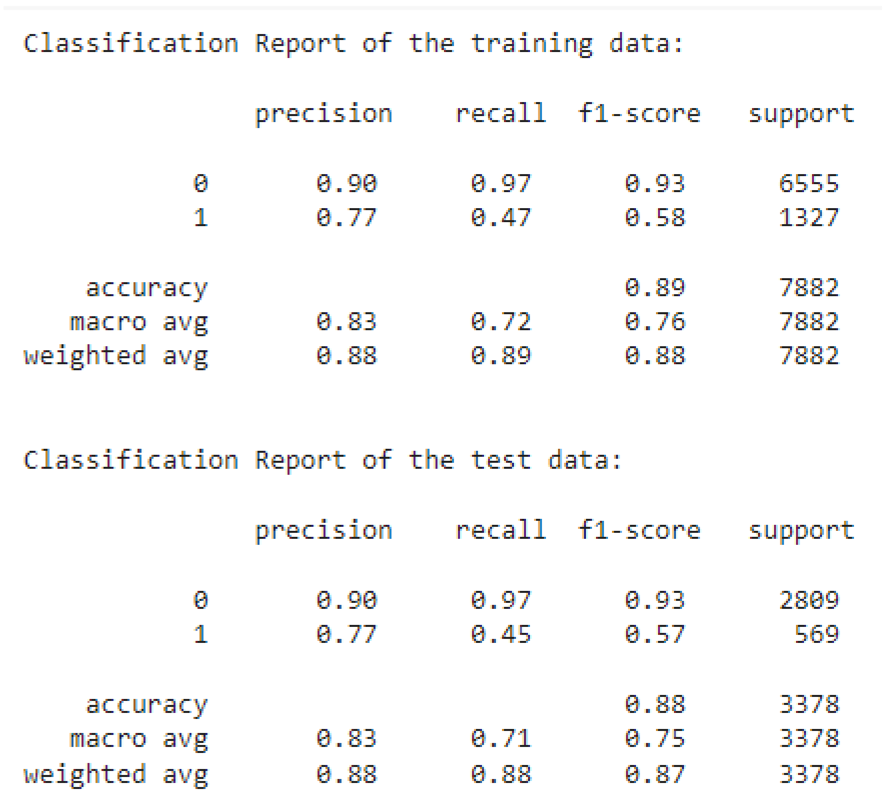
```
Accuracy score of training dataset: 0.8869576249682821

Accuracy score of testing dataset: 0.8848431024274719
```

Below is the confusion matrix obtained from this model-

```
Confusion matrix of training dataset
array([[6364,  191],
       [ 700,  627]])
```

```
Confusion matrix of testing dataset
array([[2731,   78],
       [ 311,  258]])
```

Below is the classification report obtained from this model-



```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       0.90      0.97      0.93      6555
           1       0.77      0.47      0.58      1327

    accuracy                           0.89      7882
   macro avg       0.83      0.72      0.76      7882
weighted avg       0.88      0.89      0.88      7882


Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.90      0.97      0.93      2809
           1       0.77      0.45      0.57       569

    accuracy                           0.88      3378
   macro avg       0.83      0.71      0.75      3378
weighted avg       0.88      0.88      0.87      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.866
AUC score and ROC curve for testing dataset
AUC: 0.866
```

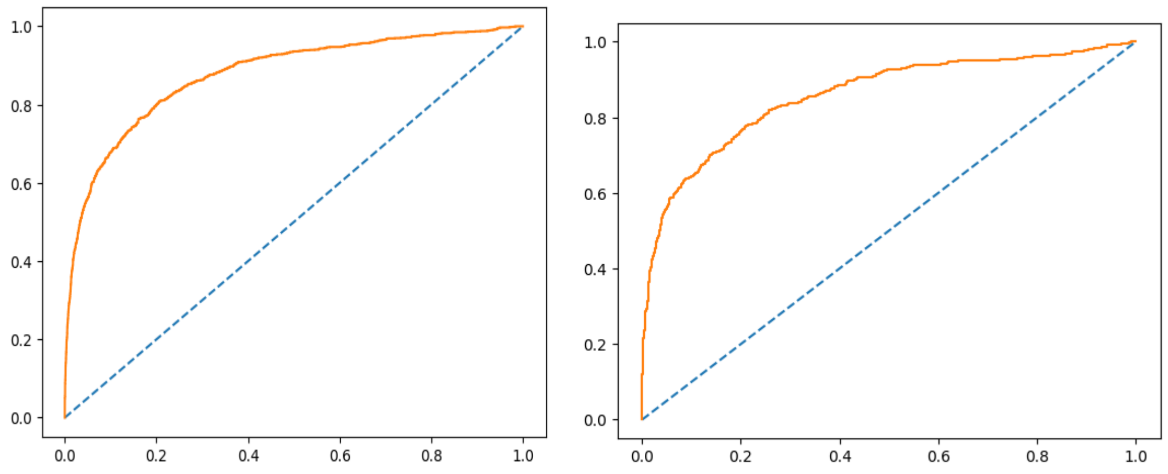


Fig. 2.1 ROC Curve and AUC Score From LDA

Below are the 10-fold cross validation scores-

```
cross validation score for training dataset
array([0.87325729, 0.89100127, 0.88071066, 0.88832487, 0.87944162,
       0.88324873, 0.87563452, 0.89593909, 0.89340102, 0.90482234])
```

```
cross validation score for testing dataset
array([0.90236686, 0.84023669, 0.85798817, 0.9112426 , 0.87573964,
       0.87573964, 0.8816568 , 0.88757396, 0.884273  , 0.87537092])
```

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

3.2 Building model using GridSearchCV and analysing the best parameters-

Using the GridSearchCV function we tried finding the best parameters to further tune-in the above model for better accuracy. 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: 0.887338239025628
```

```
Accuracy of testing dataset after gridsearchCV: 0.8848431024274719
```

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

```
confusuon matrix for training dataset
array([[6368,  187],
       [ 701,  626]])
```

```
confusuon matrix for testing dataset
array([[2733,   76],
       [ 313,  256]])
```

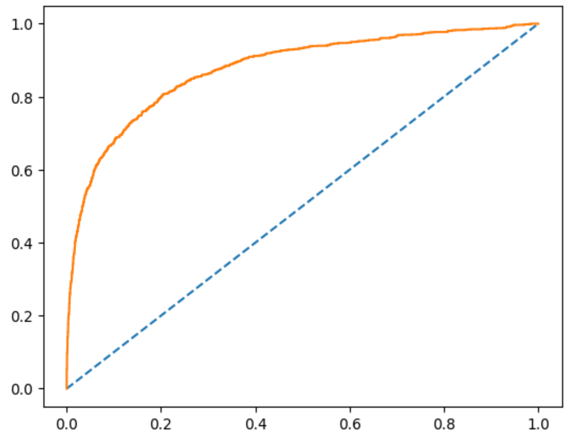
Below are the classification reports obtained from model built using GridSearchCV-

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

           0       0.90      0.97      0.93      6555
           1       0.77      0.47      0.59      1327

    accuracy                           0.89      7882
   macro avg       0.84      0.72      0.76      7882
weighted avg       0.88      0.89      0.88      7882
```

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

           0       0.90      0.97      0.93      2809
           1       0.77      0.45      0.57       569

    accuracy                           0.88      3378
   macro avg       0.83      0.71      0.75      3378
weighted avg       0.88      0.88      0.87      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.876
AUC score and ROC curve for testing dataset
AUC: 0.858
```



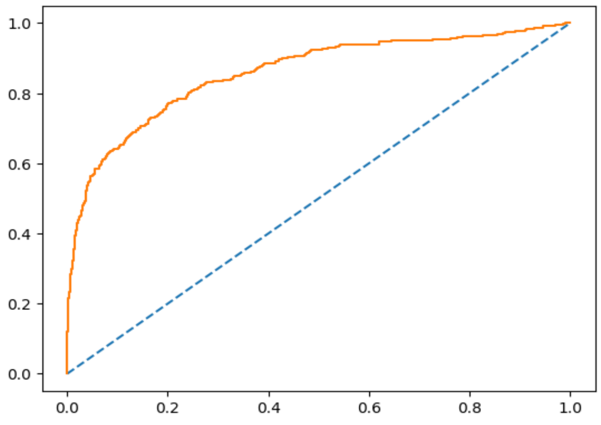


Fig.2.2 ROC Curve and AUC Score From LDA with Hypertuning

Below are the 10-fold cross validation scores-

```
cross validation scores for training dataset
array([0.87452471, 0.88846641, 0.88324873, 0.8857868 , 0.87944162,
       0.8819797 , 0.87817259, 0.89467005, 0.89340102, 0.90482234])

cross validation scores from testing dataset
array([0.90236686, 0.84023669, 0.85798817, 0.9112426 , 0.87573964,
       0.87573964, 0.8816568 , 0.88757396, 0.884273  , 0.87537092])
```

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

3.3 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 the 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: 0.8057971014492754

Accuracy of testing dataset: 0.7835997631734755
```

Below is the confusion matrix obtained from balanced dataset-

```
confusion matrix for training dataset
array([[5139, 1416],
       [1130, 5425]])
```

```
confusion matrix for testing dataset
array([[2196,  613],
       [ 118,  451]])
```

Below is the classification report obtained from balanced dataset-

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

           0       0.82      0.78      0.80      6555
           1       0.79      0.83      0.81      6555

    accuracy                           0.81     13110
   macro avg       0.81      0.81      0.81     13110
weighted avg       0.81      0.81      0.81     13110

Classification report for test dataset
              precision    recall  f1-score   support

           0       0.95      0.78      0.86      2809
           1       0.42      0.79      0.55       569

    accuracy                           0.78      3378
   macro avg       0.69      0.79      0.70      3378
weighted avg       0.86      0.78      0.81      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.888
AUC score and ROC curve for testing dataset
AUC: 0.864
```

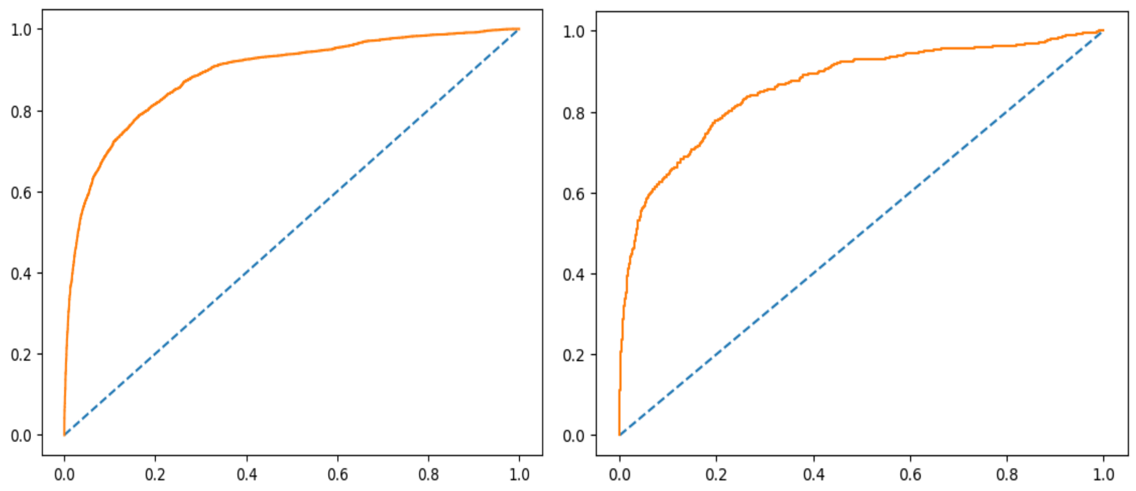


Fig.2.3 ROC Curve and AUC Score From LDA with SMOTE

Below are the 10-fold cross validation scores-

```
cross validation scores for training dataset
array([0.80015256, 0.78108314, 0.80320366, 0.80930587, 0.80472921,
       0.81388253, 0.81617086, 0.78947368, 0.81617086, 0.82227307])
```

```
cross validation scores for testing dataset
array([0.90236686, 0.84023669, 0.85798817, 0.9112426 , 0.87573964,
       0.87573964, 0.8816568 , 0.88757396, 0.884273  , 0.87537092])
```

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

Inference 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 models built using GridSearchCV. Model built on a 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.

4. Building KNN Model-

4.1 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 with default hyperparameters with default value of n_neighbour as "5".

Below are the accuracy scores obtained from this model-

```
Accracy of training dataset: 0.9545800558233951

 accuracy for testing dataset 0.9177027827116637
```

Below are the confusion matrices obtained from this model-

```
confusion matrix of training dataset
[[6432  123]
 [ 235 1092]]
```

```
confusion matrix for testing dataset
[[2715   94]
 [ 184  385]]
```

Below are the classification Report obtained from this model-

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

           0       0.96      0.98      0.97      6555
           1       0.90      0.82      0.86      1327

    accuracy                           0.95      7882
   macro avg       0.93      0.90      0.92      7882
weighted avg       0.95      0.95      0.95      7882
```

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

           0       0.94      0.97      0.95      2809
           1       0.80      0.68      0.73       569

    accuracy                           0.92      3378
   macro avg       0.87      0.82      0.84      3378
weighted avg       0.91      0.92      0.91      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.986
AUC score and ROC curve for testing dataset
AUC: 0.941
```

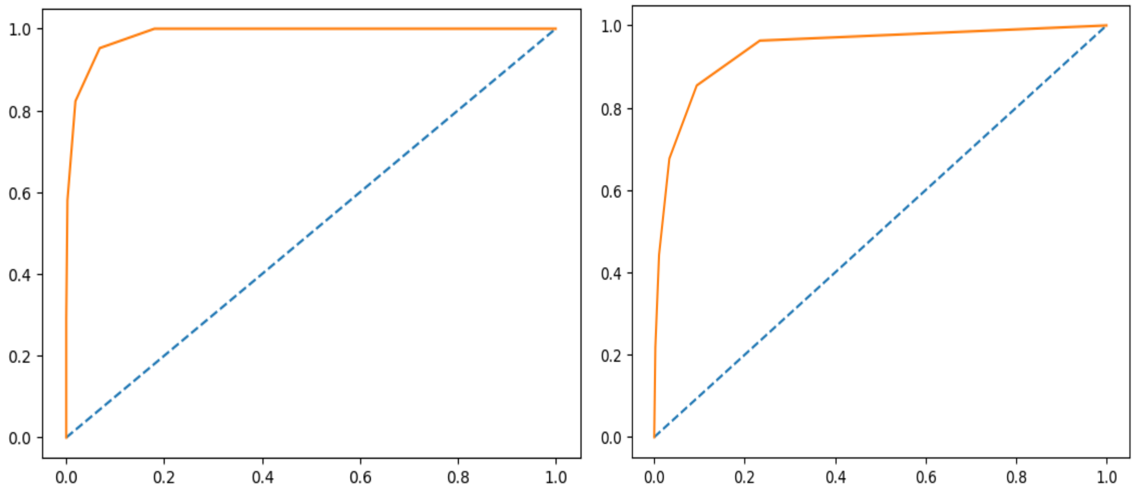


Fig.3.1 ROC Curve and AUC Scores From KNN

Observation-Train dataset has overfit because of a smaller number of neighbours selection. Let us try to observe the f1-score and accuracy for different values of K (neighbours) again using the above hyperparameters**.**

Below are the 10-fold cross validation scores-

```
cross validation scores for train dataset
array([0.89607098, 0.92141952, 0.91116751, 0.90989848, 0.91116751,
       0.90609137, 0.90228426, 0.92639594, 0.91243655, 0.91370558])

 cross validation scores for test dataset
 array([0.88757396, 0.85798817, 0.85798817, 0.88461538, 0.89349112,
        0.87278107, 0.88461538, 0.88461538, 0.83976261, 0.87240356])
```

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

4.2Tuning KNN with varying values of K-

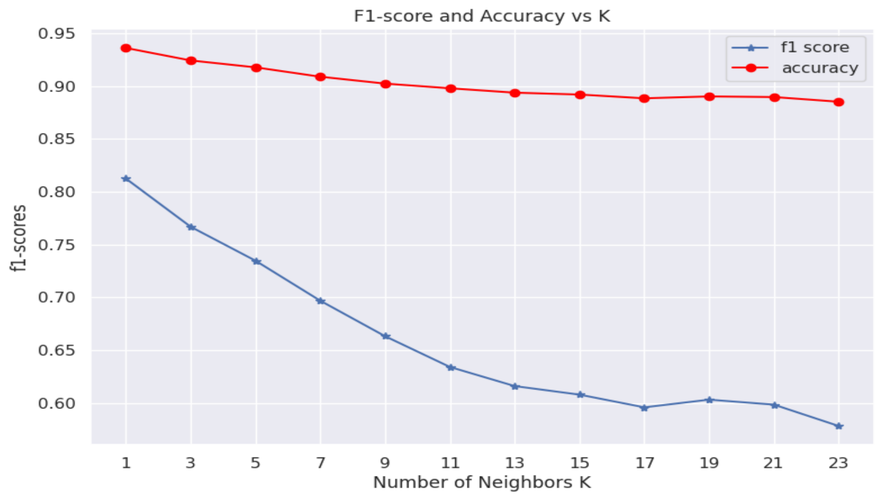


Fig.3.2 f1-score and Accuracy vs K

Observation-

- 5 seems to be the optimum value but for that, train dataset has fully grown.
- KNN_model2 has the best grid with best neighbours. Even though the training data performance has fully grown, test data is not far behind and has a difference of 7%. Let's use cross validation on full data to see if the f1-score of 93% holds true. For 5-fold cross validation on full data, the following are the F1-scores.

4.3 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_neoghbors value.

Below are the MSE scores-

```
[0.06394316163410307,
 0.07578448786264058,
 0.08229721728833628,
 0.09117821195973952,
 0.09769094138543521,
 0.10213143872113672,
 0.10627590290112487,
 0.10805210183540559,
 0.11160449970396668,
 0.10982830076968621]
```

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

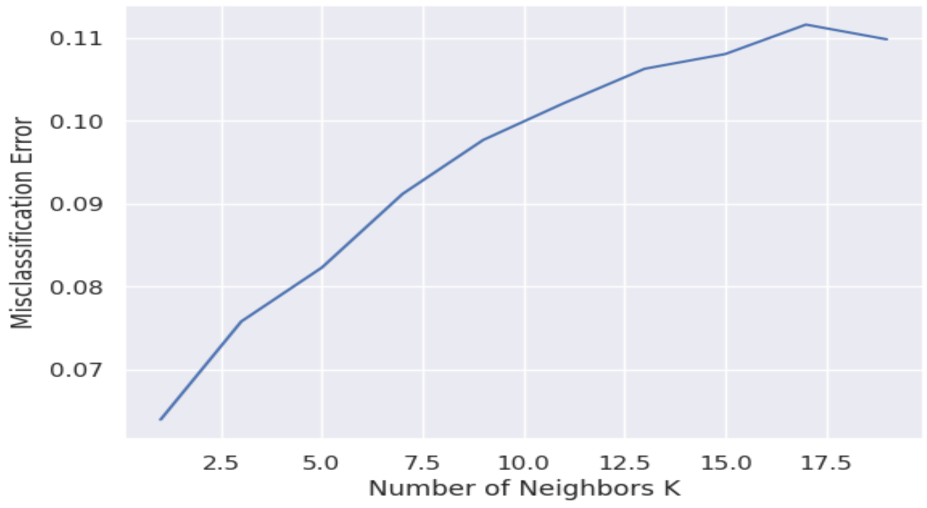


Fig.3.3 Graphical Version Of MSE Score

4.4 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 hyperparameters to check if we can outperform the accuracy achieved by the model built with default values of the hyperparameter. From GridSearchCV we found that the best parameters are “ball-tree” as algorithm, “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: 1.0

Accuracy of testing dataset after gridsearchCV: 0.9547069271758437
```

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

```
confusuon matrix for training dataset
array([[6555,    0],
       [   0, 1327]])
```

```
confusuon matrix for testing dataset
array([[2757,   52],
       [ 101,  468]])
```

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

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

           0       1.00      1.00      1.00      6555
           1       1.00      1.00      1.00      1327

    accuracy                           1.00      7882
   macro avg       1.00      1.00      1.00      7882
weighted avg       1.00      1.00      1.00      7882
```

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

           0       0.96      0.98      0.97      2809
           1       0.90      0.82      0.86       569

    accuracy                           0.95      3378
   macro avg       0.93      0.90      0.92      3378
weighted avg       0.95      0.95      0.95      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 1.000
AUC score and ROC curve for testing dataset
AUC: 0.979
```

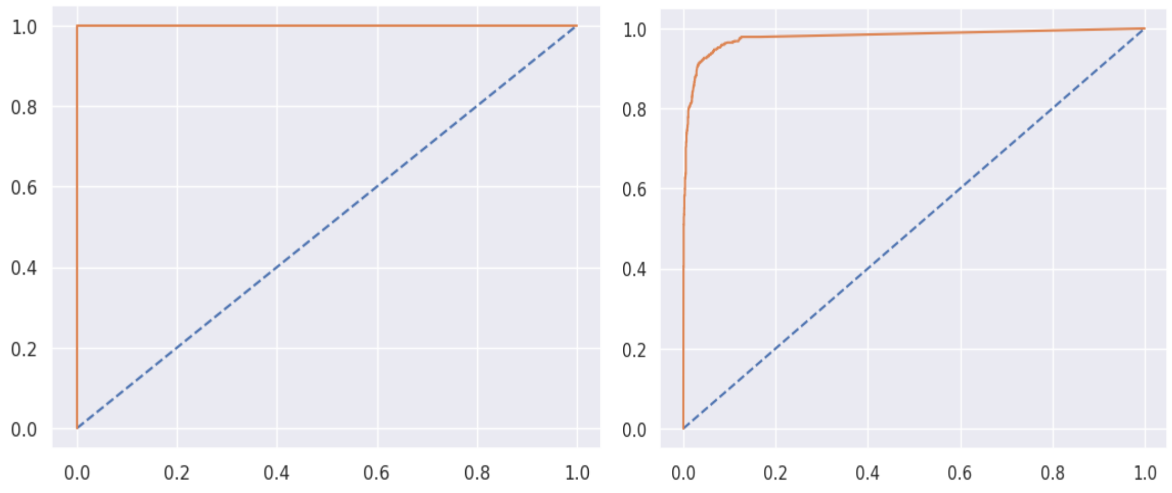


Fig.3.4 ROC Curve and AUC Score From KNN with Hyperparameter Tuning

Below are the 10-fold cross validation scores-

```
cross validation scores for train dataset
array([0.93789607, 0.95437262, 0.95939086, 0.95431472, 0.95812183,
       0.96573604, 0.94162437, 0.96700508, 0.95304569, 0.95939086])
```

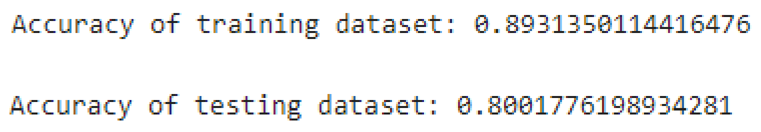
```
cross validation scores for test dataset
array([0.90828402, 0.8816568 , 0.89940828, 0.90236686, 0.9260355 ,
       0.91420118, 0.93195266, 0.91715976, 0.90207715, 0.9347181 ])
```

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

4.5 Building KNN model over balanced dataset using SMOTE-

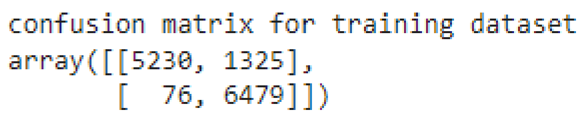
From above descriptive analysis we can conclude that the original data provided is imbalance in nature and by using SMOTE technique we can balance the data to check if the model can outperform when data is balanced. We have applied the SMOTE technique to oversample the data and to obtain a balanced dataset.

Below are the accuracy scores obtained from balanced dataset-

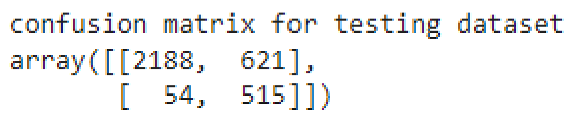


```
Accuracy of training dataset: 0.8931350114416476

Accuracy of testing dataset: 0.8001776198934281
```

Below are the confusion matrices obtained from balanced dataset-

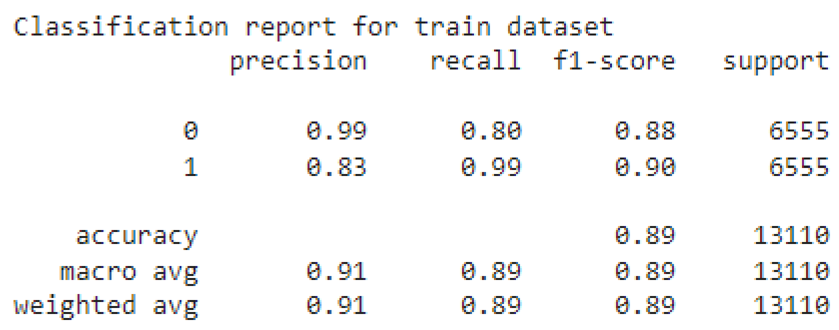


```
confusion matrix for training dataset
array([[5230, 1325],
       [  76, 6479]])
```

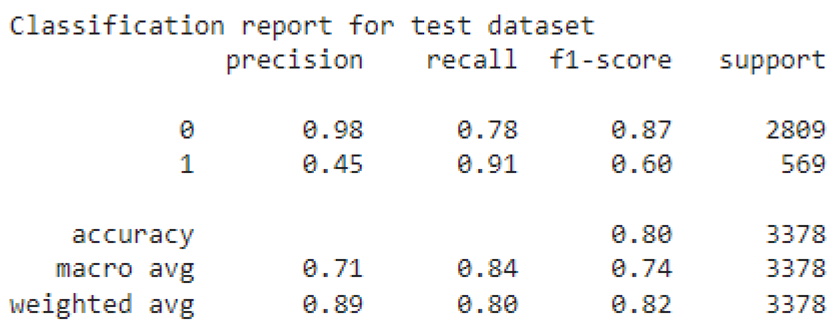


```
confusion matrix for testing dataset
array([[2188,  621],
       [  54,  515]])
```

Below are the classification reports obtained from balanced dataset-



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

           0       0.99      0.80      0.88      6555
           1       0.83      0.99      0.90      6555

    accuracy                           0.89     13110
   macro avg       0.91      0.89      0.89     13110
weighted avg       0.91      0.89      0.89     13110
```



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

           0       0.98      0.78      0.87      2809
           1       0.45      0.91      0.60       569

    accuracy                           0.80      3378
   macro avg       0.71      0.84      0.74      3378
weighted avg       0.89      0.80      0.82      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.984
AUC score and ROC curve for testing dataset
AUC: 0.928
```

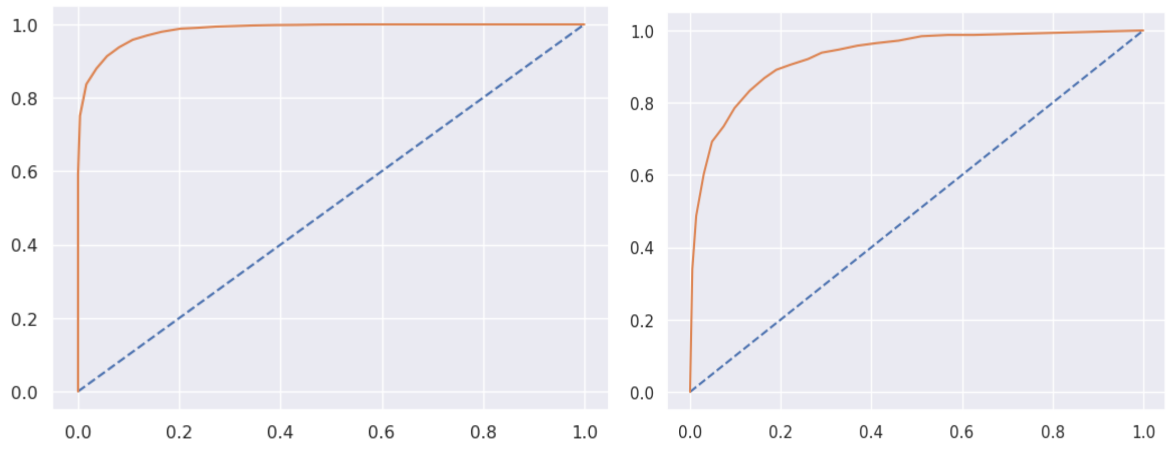


Fig.3.5 ROC Curve and AUC Scores From KNN with SMOTE

Below are the 10-fold cross validation scores-

```
cross validation scores for train dataset
array([0.86575133, 0.85583524, 0.90541571, 0.86956522, 0.8733791 ,
       0.8863463 , 0.87795576, 0.88024409, 0.86880244, 0.89016018])

cross validation scores for test dataset
array([0.88757396, 0.84319527, 0.84023669, 0.86390533, 0.87869822,
       0.87573964, 0.87573964, 0.87573964, 0.84866469, 0.88724036])
```

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

Inference-

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 the 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 a 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.

5. Building Gaussian Naive Bayes-

5.1 Building model with default hyperparameters-

Post splitting data into training and testing we are now ready to build model using Naive Bayes algorithm. Bayes' algorithm is based on Bayes theorem of conditional probability. Consider 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: 0.7854605430093885

Accracy of testing dataset: 0.7818235642391947
```

Below are confusion matrices and classification reports using this model-

```
Confusion matrix of train dataset
[[5240 1315]
 [ 376  951]]
              precision    recall  f1-score   support

           0       0.93      0.80      0.86      6555
           1       0.42      0.72      0.53      1327

    accuracy                           0.79      7882
   macro avg       0.68      0.76      0.70      7882
weighted avg       0.85      0.79      0.81      7882

Confusion matrix of test dataset
[[2234  575]
 [ 162  407]]
Classification report of test dataset
              precision    recall  f1-score   support

           0       0.93      0.80      0.86      2809
           1       0.41      0.72      0.52       569

    accuracy                           0.78      3378
   macro avg       0.67      0.76      0.69      3378
weighted avg       0.85      0.78      0.80      3378
```

Below are AUC scores and ROC curves using this model-

```
AUC score and ROC curve for training dataset
AUC: 0.827
AUC score and ROC curve for testing dataset
AUC: 0.810
```

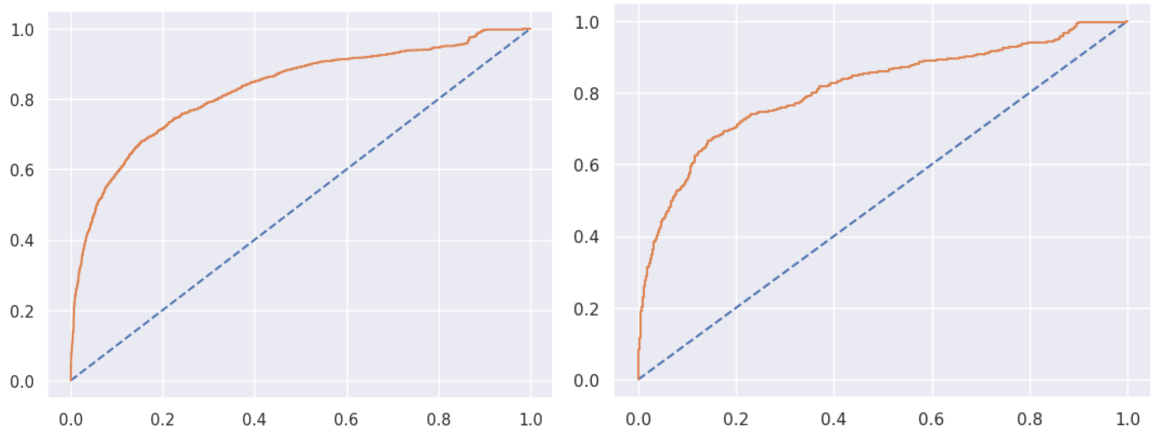


Fig.4.1 ROC Curve and AUC Scores From Naïve Bayes

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

```
cross validation scores for train dataset
array([0.76299113, 0.7896071 , 0.76395939, 0.79187817, 0.7677665 ,
       0.79187817, 0.80203046, 0.77918782, 0.79187817, 0.80329949])
```

```
cross validation scores for test dataset
array([0.80473373, 0.73668639, 0.72781065, 0.81065089, 0.76331361,
       0.78402367, 0.76627219, 0.80769231, 0.71513353, 0.78635015])
```

5.2 Building Gaussian Naive Bayes over balanced data using SMOTE-

After building a naive 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 Naive Bayes algorithm over balanced dataset-

```
Accracy of training dataset: 0.736231884057971
```

```
Accracy of testing dataset: 0.6397276494967437
```

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

```
Confusion matrix of train dataset
[[3970 2585]
 [ 873 5682]]
              precision    recall  f1-score   support

           0       0.82      0.61      0.70      6555
           1       0.69      0.87      0.77      6555

    accuracy                           0.74     13110
   macro avg       0.75      0.74      0.73     13110
weighted avg       0.75      0.74      0.73     13110
```

```
Confusion matrix of test dataset
[[1683 1126]
 [  91  478]]
Classification report of test dataset
              precision    recall  f1-score   support

           0       0.95      0.60      0.73      2809
           1       0.30      0.84      0.44       569

    accuracy                           0.64      3378
   macro avg       0.62      0.72      0.59      3378
weighted avg       0.84      0.64      0.68      3378
```

Below are the AUC scores and ROC curves obtained using Naïve Bayes algorithm over balanced dataset-

```
AUC score and ROC curve for training dataset
AUC: 0.837
```

```
AUC score and ROC curve for testing dataset
AUC: 0.809
```

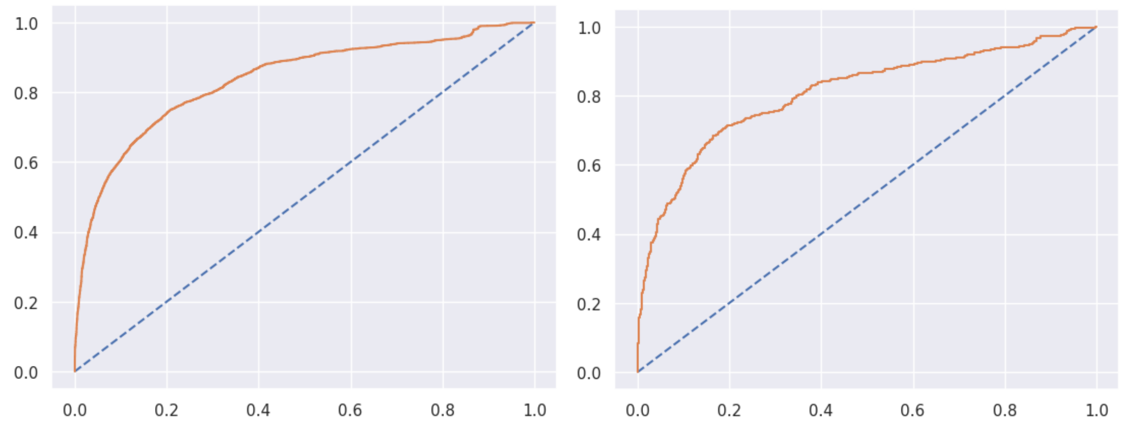


Fig.4.2 ROC Curve and AUC Scores From Naïve Bayes with SMOTE

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

```
cross validation scores for train dataset
array([0.72158658, 0.72234935, 0.74065599, 0.74141876, 0.72158658,
       0.74065599, 0.75133486, 0.71700992, 0.74599542, 0.75591152])

cross validation scores for test dataset
array([0.80473373, 0.73668639, 0.72781065, 0.81065089, 0.76331361,
       0.78402367, 0.76627219, 0.80769231, 0.71513353, 0.78635015])
```

Inference from Naive 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 the 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 models built on a balanced dataset. Model built on a balanced 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.

6. Ensemble Model-Random Forest-

Random forest is an ensemble machine learning algorithm that uses bootstrapping to reduce variance in the underlying decision trees. It also selects only a subset of features for each node split decision. Since it is an ensemble of trees with varying features for each node, each tree is different from another. Random forest is resistant to outliers and does not require the data to be scaled.

6.1 Random Forest base model with default hyperparameters-

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

```
accuracy score for training dataset: 1.0
confusion matrix for training dataset
[[6555    0]
 [   0 1327]]
classification report for training dataset
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      6555
           1       1.00      1.00      1.00      1327

    accuracy                           1.00      7882
   macro avg       1.00      1.00      1.00      7882
weighted avg       1.00      1.00      1.00      7882
```

```
accuracy score for testing dataset: 0.9742451154529307
confusion matrix for testing dataset
[[2794   15]
 [  72  497]]
classification report for testing dataste
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      2809
           1       0.97      0.87      0.92       569

    accuracy                           0.97      3378
   macro avg       0.97      0.93      0.95      3378
weighted avg       0.97      0.97      0.97      3378
```

Below are ROC Curve and AUC Score of train and test dataset-

```
AUC score and ROC curve for training dataset
AUC: 0.825
AUC score and ROC curve for testing dataset
AUC: 0.809
```

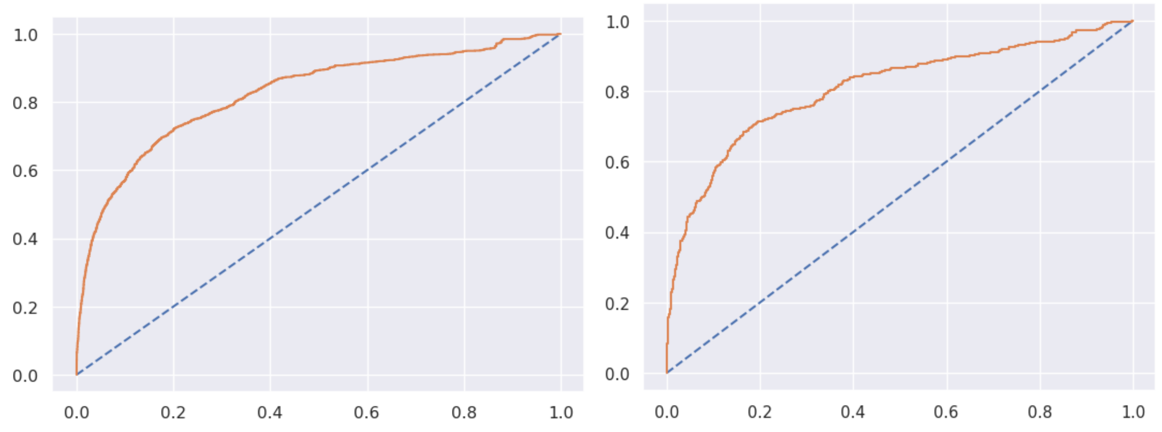
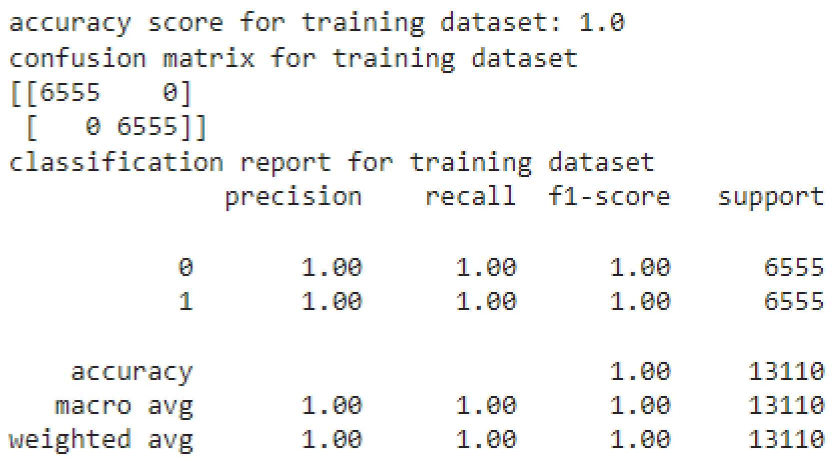


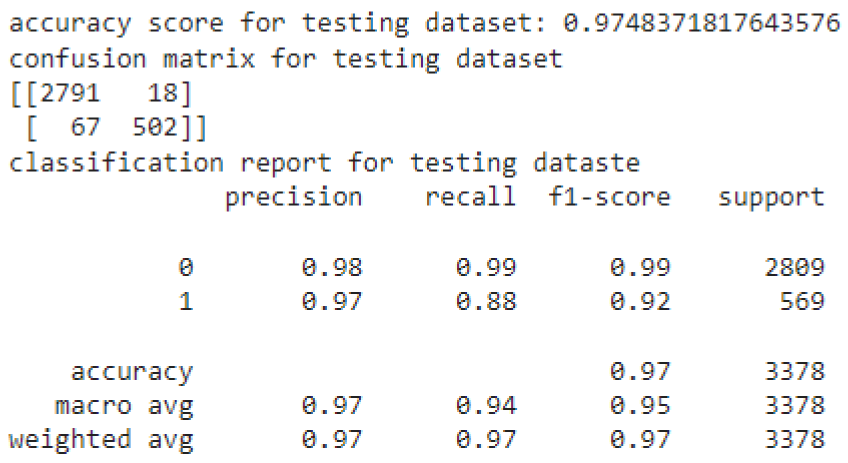
Fig.5.1 ROC Curve and AUC Scores From Random Forest

6.2 Building random forest using GridSearchCV and getting the best hyperparameters-

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



```
accuracy score for training dataset: 1.0
confusion matrix for training dataset
[[6555    0]
 [   0 6555]]
classification report for training dataset
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      6555
           1       1.00      1.00      1.00      6555

    accuracy                           1.00     13110
   macro avg       1.00      1.00      1.00     13110
weighted avg       1.00      1.00      1.00     13110
```



```
accuracy score for testing dataset: 0.9748371817643576
confusion matrix for testing dataset
[[2791   18]
 [  67  502]]
classification report for testing dataste
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      2809
           1       0.97      0.88      0.92       569

    accuracy                           0.97      3378
   macro avg       0.97      0.94      0.95      3378
weighted avg       0.97      0.97      0.97      3378
```

6.3 Feature Importance-

This model has overfitted both in the default base model as well as the hyper parameter tuned model. The tuned model in fact performed worse than the base model. Hence in the final comparison, this cannot get selected as the best model.

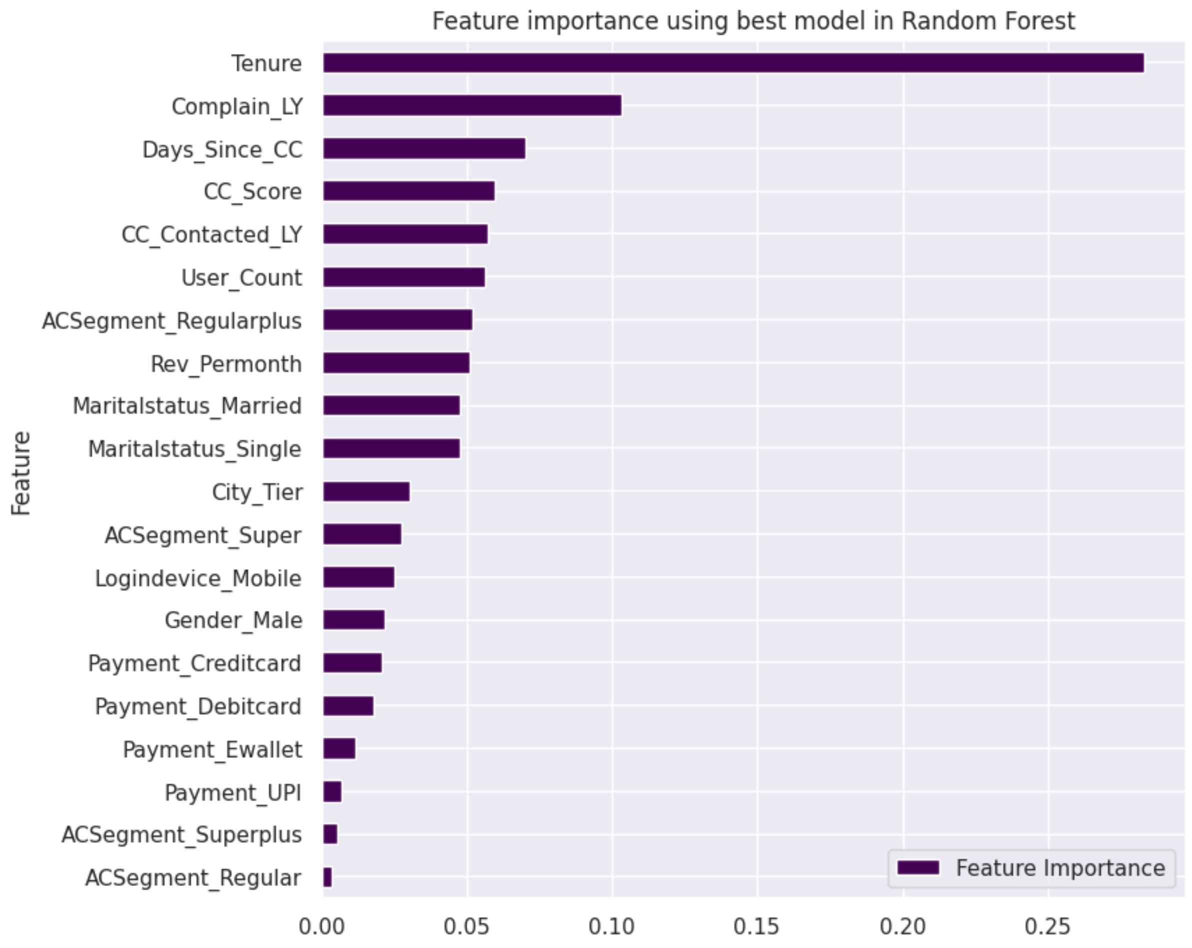


Fig.5.2 Feature importance from the best Random Forest model

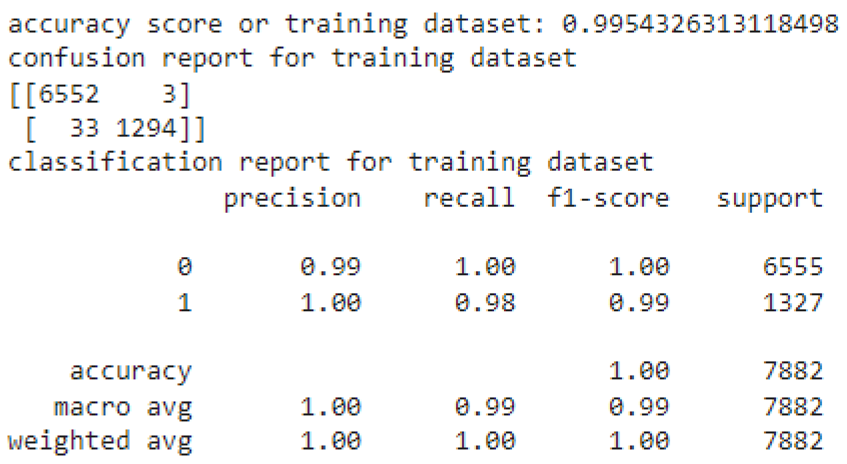
This model has picked Tenure to be the most important feature followed by Customer care contacted previous year and Days since customer care last contact.

- From EDA, we can observe that for lower tenures especially within the first year, the churn is higher. Hence, once a customer has been acquired, the first year is very important to keep the customer satisfied.
- The next important parameter to predict customer churn per this model is the number of times customer care was contacted by the customer. Per EDA, the median and third quartile of number of times customer care was contacted previous year is higher for churned customers compared to active/current customers.
- Churned customers had contacted customer care more recently before churning than active customers. Per EDA, median days since last customer connect is higher for active customers. Churned customers had contacted customer care recently before churning.

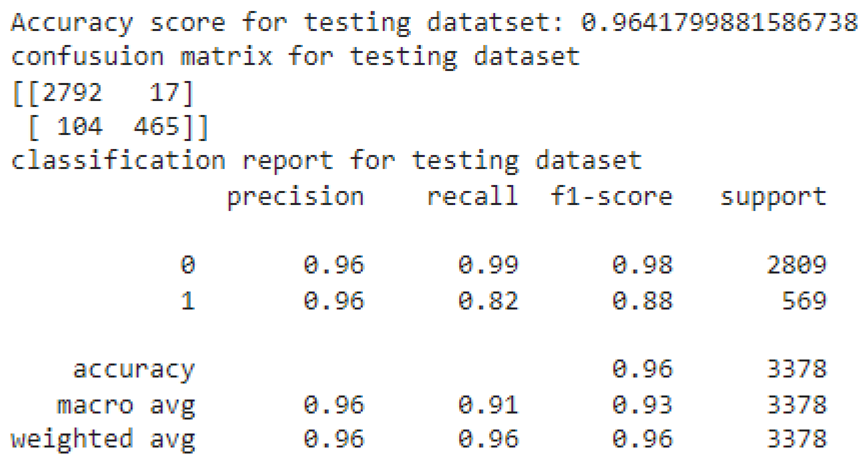
7.1 Bagging on original dataset-

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

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



```
accuracy score or training dataset: 0.9954326313118498
confusion report for training dataset
[[6552    3]
 [  33 1294]]
classification report for training dataset
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      6555
           1       1.00      0.98      0.99      1327

    accuracy                           1.00      7882
   macro avg       1.00      0.99      0.99      7882
weighted avg       1.00      1.00      1.00      7882
```



```
Accuracy score for testing datatset: 0.9641799881586738
confusuion matrix for testing dataset
[[2792   17]
 [ 104  465]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.96      0.99      0.98      2809
           1       0.96      0.82      0.88       569

    accuracy                           0.96      3378
   macro avg       0.96      0.91      0.93      3378
weighted avg       0.96      0.96      0.96      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 1.000
AUC score and ROC curve for testing dataset
AUC: 0.990
```

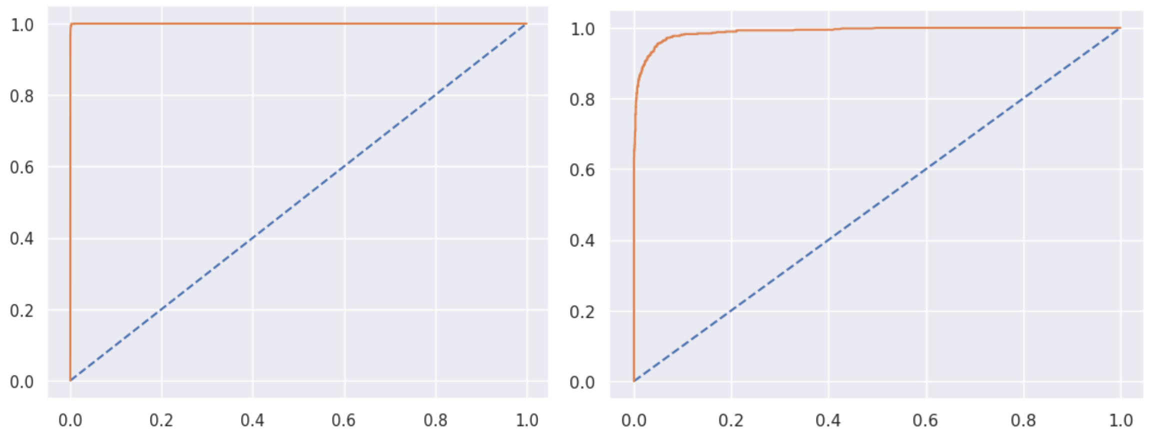


Fig.6.1 ROC Curve and AUC Score Bagging On Original Dataset

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

```
array([0.95690748, 0.97338403, 0.96954315, 0.95939086, 0.96192893,
       0.97208122, 0.94923858, 0.9606599 , 0.95939086, 0.95812183])
array([0.94674556, 0.91420118, 0.91715976, 0.9408284 , 0.92307692,
       0.93786982, 0.9260355 , 0.93195266, 0.91691395, 0.94065282])
```

7.2 Bagging on Balanced dataset-

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

```
accuracy score or training dataset: 0.998093058733791
confusion report for training dataset
[[6547    8]
 [  17 6538]]
classification report for training dataset
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      6555
           1       1.00      1.00      1.00      6555

    accuracy                           1.00     13110
   macro avg       1.00      1.00      1.00     13110
weighted avg       1.00      1.00      1.00     13110
```

```
Accuracy score for testing datatset: 0.9656601539372409
confusuion matrix for testing dataset
[[2771   38]
 [  78  491]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      2809
           1       0.93      0.86      0.89       569

    accuracy                           0.97      3378
   macro avg       0.95      0.92      0.94      3378
weighted avg       0.97      0.97      0.97      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 1.000
AUC score and ROC curve for testing dataset
AUC: 0.988
```

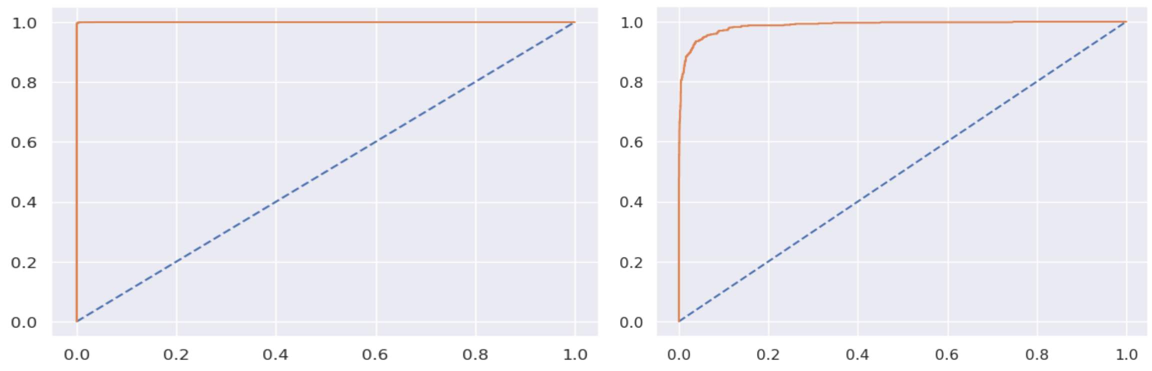


Fig.6.2 ROC Curve and AUC Scores from Bagging On Balanced Dataset

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

cross validation scores for test dataset-

```
array([0.93897788, 0.94050343, 0.99237223, 0.99313501, 0.98855835,
       0.98855835, 0.98932113, 0.99160946, 0.99237223, 0.99237223])
```

cross validation scores for test dataset-

```
array([0.94674556, 0.91420118, 0.91715976, 0.9408284 , 0.92307692,
       0.93786982, 0.9260355 , 0.93195266, 0.91691395, 0.94065282])
```

8. Ensemble Method AdaBoost-

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

8.1 Building Ada-Boost Model on original dataset-

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

```
Accuracy for training dataset: 0.8987566607460036
confusion matrix for training dataset
[[6276  279]
 [ 519  808]]
classification report for training dataset
              precision    recall  f1-score   support

           0       0.92      0.96      0.94      6555
           1       0.74      0.61      0.67      1327

    accuracy                           0.90      7882
   macro avg       0.83      0.78      0.80      7882
weighted avg       0.89      0.90      0.89      7882
```

```
accuracy score for testing dataset: 0.8996447602131439
confusion matrix for testing dataset
[[2698  111]
 [ 228  341]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.92      0.96      0.94      2809
           1       0.75      0.60      0.67       569

    accuracy                           0.90      3378
   macro avg       0.84      0.78      0.80      3378
weighted avg       0.89      0.90      0.89      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.915
AUC score and ROC curve for testing dataset
AUC: 0.903
```

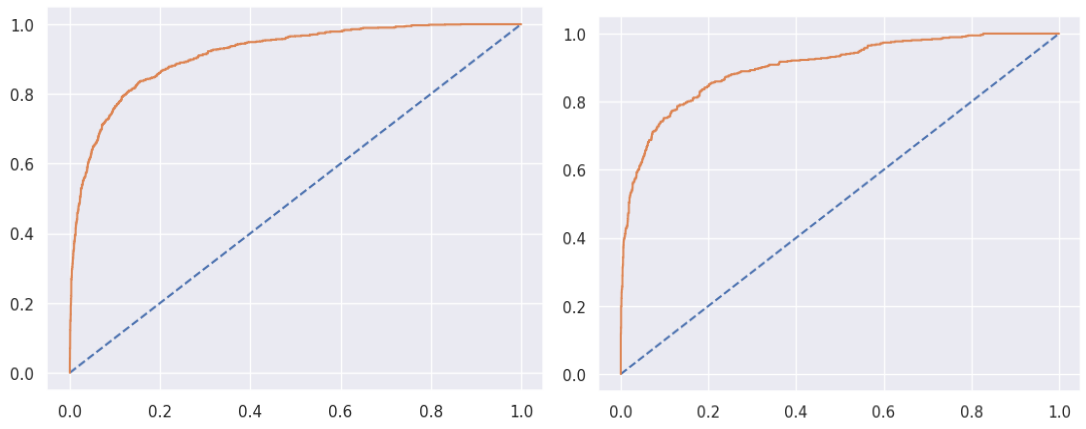
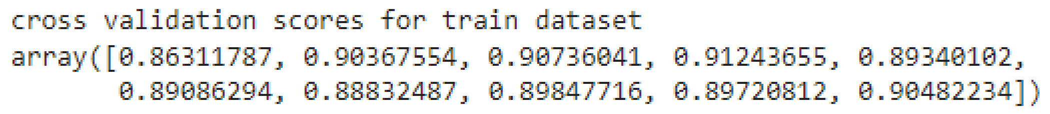


Fig. 7.1 ROC curve and AUC score from Ada-Boosting on Original Dataset

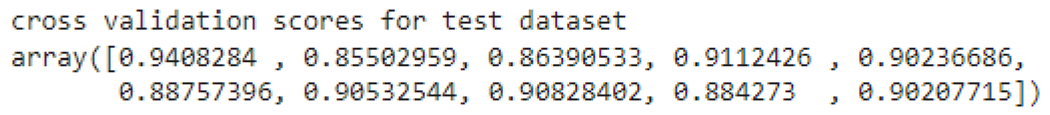
Below are 10-fold cross validation scores using this model-

Cross validation scores for test dataset-



```
cross validation scores for train dataset
array([0.86311787, 0.90367554, 0.90736041, 0.91243655, 0.89340102,
       0.89086294, 0.88832487, 0.89847716, 0.89720812, 0.90482234])
```

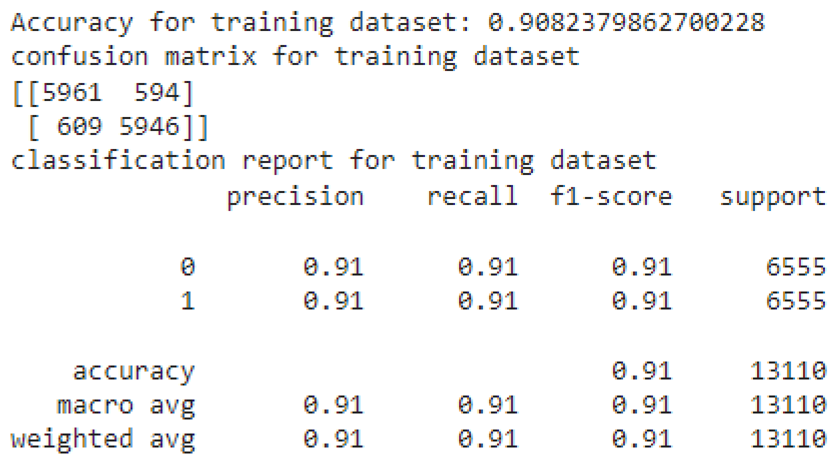
Cross validation scores for training dataset-



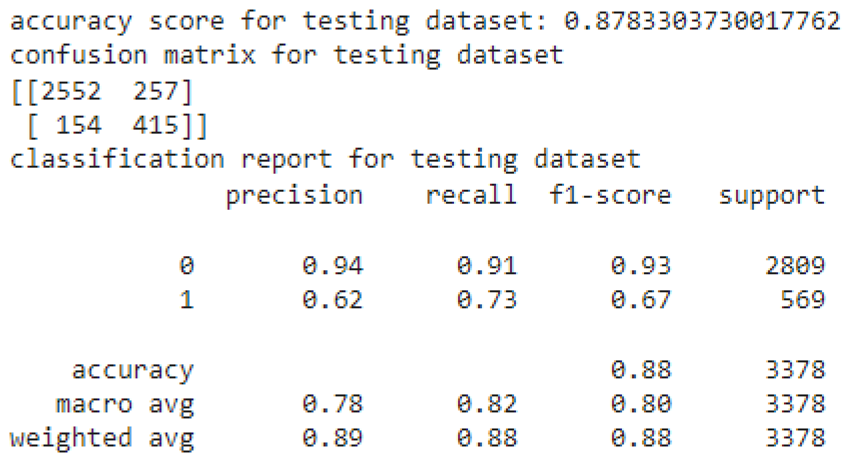
```
cross validation scores for test dataset
array([0.9408284 , 0.85502959, 0.86390533, 0.9112426 , 0.90236686,
       0.88757396, 0.90532544, 0.90828402, 0.884273  , 0.90207715])
```

8.2 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: 0.9082379862700228
confusion matrix for training dataset
[[5961  594]
 [ 609 5946]]
classification report for training dataset
              precision    recall  f1-score   support

           0       0.91      0.91      0.91      6555
           1       0.91      0.91      0.91      6555

    accuracy                           0.91     13110
   macro avg       0.91      0.91      0.91     13110
weighted avg       0.91      0.91      0.91     13110
```



```
accuracy score for testing dataset: 0.8783303730017762
confusion matrix for testing dataset
[[2552  257]
 [ 154  415]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.94      0.91      0.93      2809
           1       0.62      0.73      0.67       569

    accuracy                           0.88      3378
   macro avg       0.78      0.82      0.80      3378
weighted avg       0.89      0.88      0.88      3378
```

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

AUC score and ROC curve for training dataset

AUC: 0.967

```
AUC score and ROC curve for testing dataset
AUC: 0.901
```

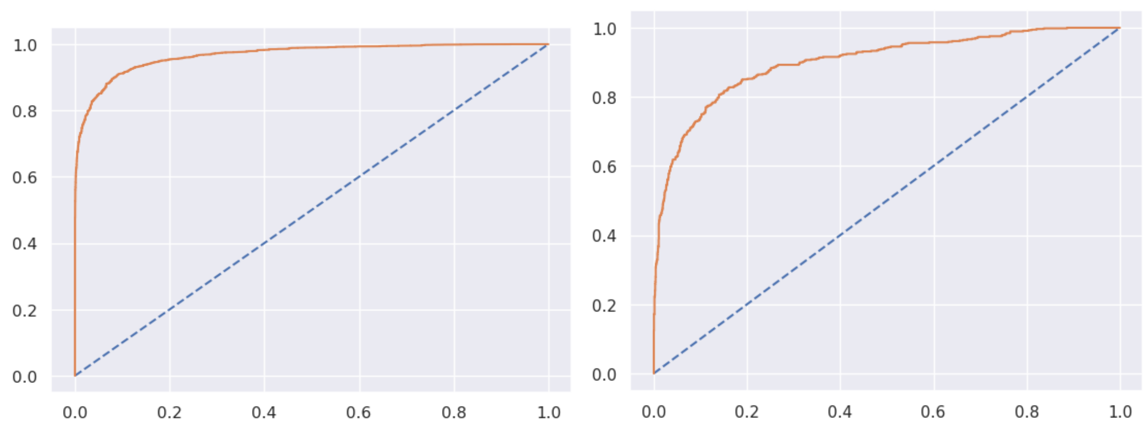


Fig.7.2 ROC Curve and AUC Score from Ada-Boosting on Balanced Dataset

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

Cross validation scores for train dataset-

```
cross validation scores for train dataset
array([0.80549199, 0.81083143, 0.92372235, 0.93058734, 0.92677346,
       0.91914569, 0.91685736, 0.91990847, 0.91914569, 0.92143402])
```

Cross validation scores for test dataset-

```
cross validation scores for test dataset
array([0.9408284 , 0.85502959, 0.86390533, 0.9112426 , 0.90236686,
       0.88757396, 0.90532544, 0.90828402, 0.884273  , 0.90207715])
```

The performance of this model when compared with other models is low. But it has not shown any overfitting or underfitting.

Feature importance from Adaboost-

As we can see in below fig.7.3 Feature importance from best Adaboost model- This model has picked User count to be the most important feature followed by Marital Status married and Tenure care contacted previous year.

- Churned customers had contacted customer care more recently before churning than active customers. Per EDA, median days since last customer connect is higher for active customers. Churned customers had contacted customer care recently before churning.
- The next important parameter to predict customer churn per this model is the number of times customer care was contacted by the customer. Per EDA, the median and third quartile of number of times customer care was contacted previous year is higher for churned customers compared to active/current customers.

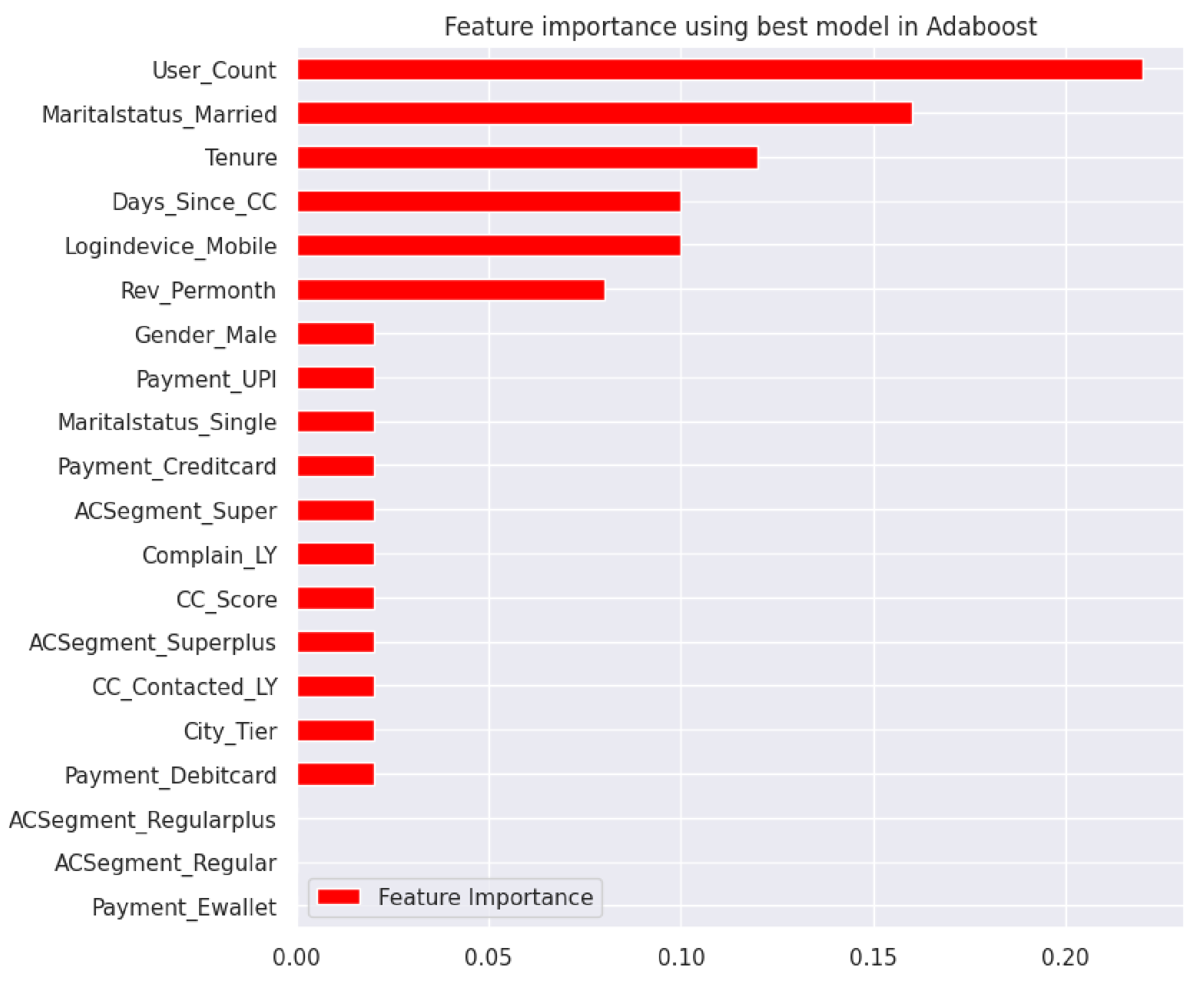


Fig. 7.3 Feature importance from best Adaboost model

9. Gradient Boosting-

Gradient boost is an ensemble machine learning algorithm that trains underlying models in a gradual, additive and sequential manner.

9.1 Building Gradient Boosting Model on original dataset-

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

```
accuracy for training dataset: 0.9222278609489977
confusion matrix for training dataset
[[6378  177]
 [ 436  891]]
classification report for training dataset
              precision    recall  f1-score   support

           0       0.94      0.97      0.95      6555
           1       0.83      0.67      0.74      1327

    accuracy                           0.92      7882
   macro avg       0.89      0.82      0.85      7882
weighted avg       0.92      0.92      0.92      7882

accuracy score for testing dataset: 0.911190053285968
confusuon matrix for testing dataset
[[2730   79]
 [ 221  348]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.93      0.97      0.95      2809
           1       0.81      0.61      0.70       569

    accuracy                           0.91      3378
   macro avg       0.87      0.79      0.82      3378
weighted avg       0.91      0.91      0.91      3378
```

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

```
AUC score and ROC curve for training dataset
AUC: 0.948
AUC score and ROC curve for testing dataset
AUC: 0.927
```

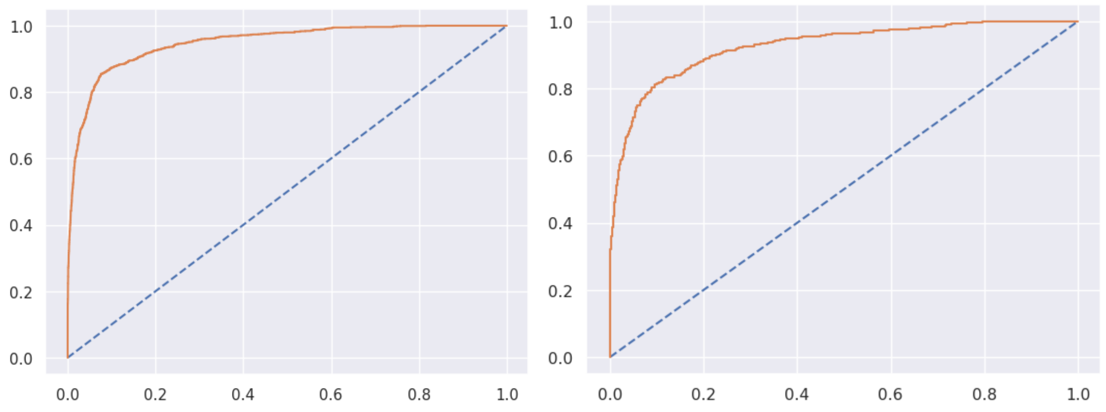
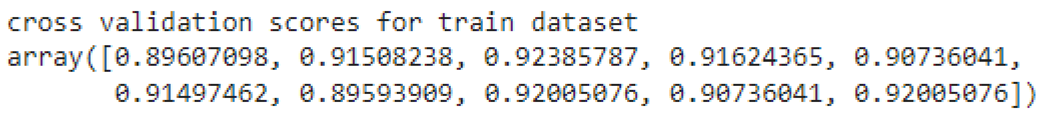


Fig.8.1 ROC Curve and AUC Score from Gradient-Boosting on Original Dataset

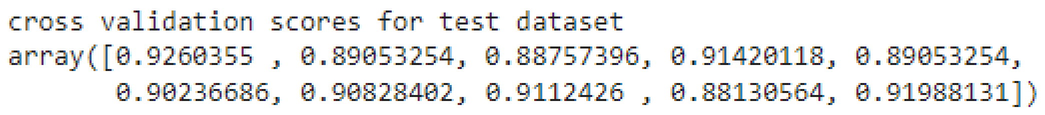
Below are 10-fold cross validation scores using this model-

Cross validation scores for train dataset-



```
cross validation scores for train dataset
array([0.89607098, 0.91508238, 0.92385787, 0.91624365, 0.90736041,
       0.91497462, 0.89593909, 0.92005076, 0.90736041, 0.92005076])
```

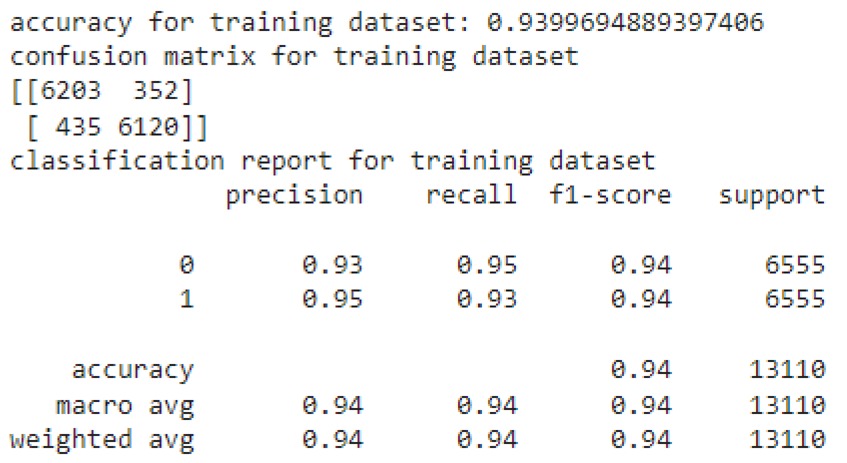
Cross validation scores for test dataset-



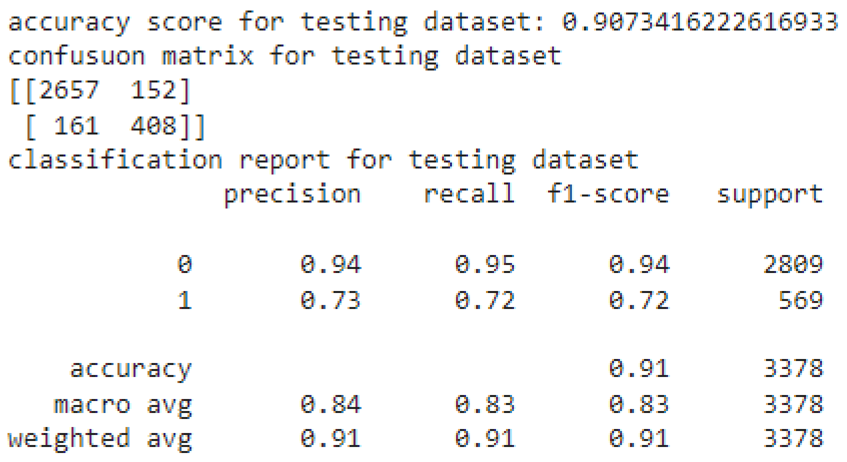
```
cross validation scores for test dataset
array([0.9260355 , 0.89053254, 0.88757396, 0.91420118, 0.89053254,
       0.90236686, 0.90828402, 0.9112426 , 0.88130564, 0.91988131])
```

9.2 Gradient Boost best model with tuned hyperparameters-

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



```
accuracy for training dataset: 0.9399694889397406
confusion matrix for training dataset
[[6203  352]
 [ 435 6120]]
classification report for training dataset
              precision    recall  f1-score   support

           0       0.93      0.95      0.94      6555
           1       0.95      0.93      0.94      6555

    accuracy                           0.94     13110
   macro avg       0.94      0.94      0.94     13110
weighted avg       0.94      0.94      0.94     13110
```



```
accuracy score for testing dataset: 0.9073416222616933
confusuon matrix for testing dataset
[[2657  152]
 [ 161  408]]
classification report for testing dataset
              precision    recall  f1-score   support

           0       0.94      0.95      0.94      2809
           1       0.73      0.72      0.72       569

    accuracy                           0.91      3378
   macro avg       0.84      0.83      0.83      3378
weighted avg       0.91      0.91      0.91      3378
```

The model is robust (no overfit or underfit) but the recall is quite poor in both train and test data. Tuning was done to see if performance can be improved on the model.

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

```
AUC score and ROC curve for training dataset
AUC: 0.984
AUC score and ROC curve for testing dataset
AUC: 0.919
```

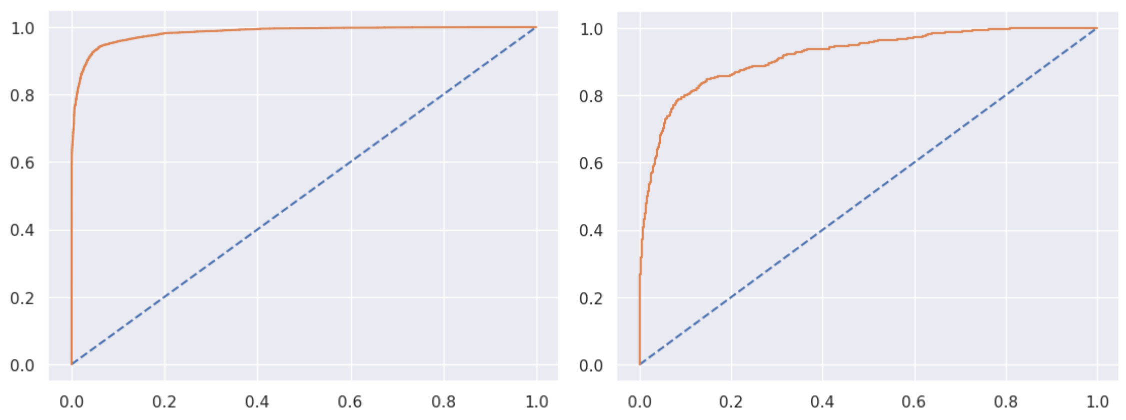


Fig. 8.2 ROC Curve and AUC Score from Gradient-Boosting

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

Cross validation scores for train dataset-

```
cross validation scores for train dataset
array([0.83371472, 0.83142639, 0.95194508, 0.95728452, 0.96033562,
       0.94965675, 0.94736842, 0.95881007, 0.95499619, 0.96338673])
```

Cross validation scores for test dataset-

```
cross validation scores for test dataset
array([0.9260355 , 0.89053254, 0.88757396, 0.91420118, 0.89053254,
       0.90236686, 0.90828402, 0.9112426 , 0.88130564, 0.91988131])
```

Feature Importance for Gradient-Boosting-

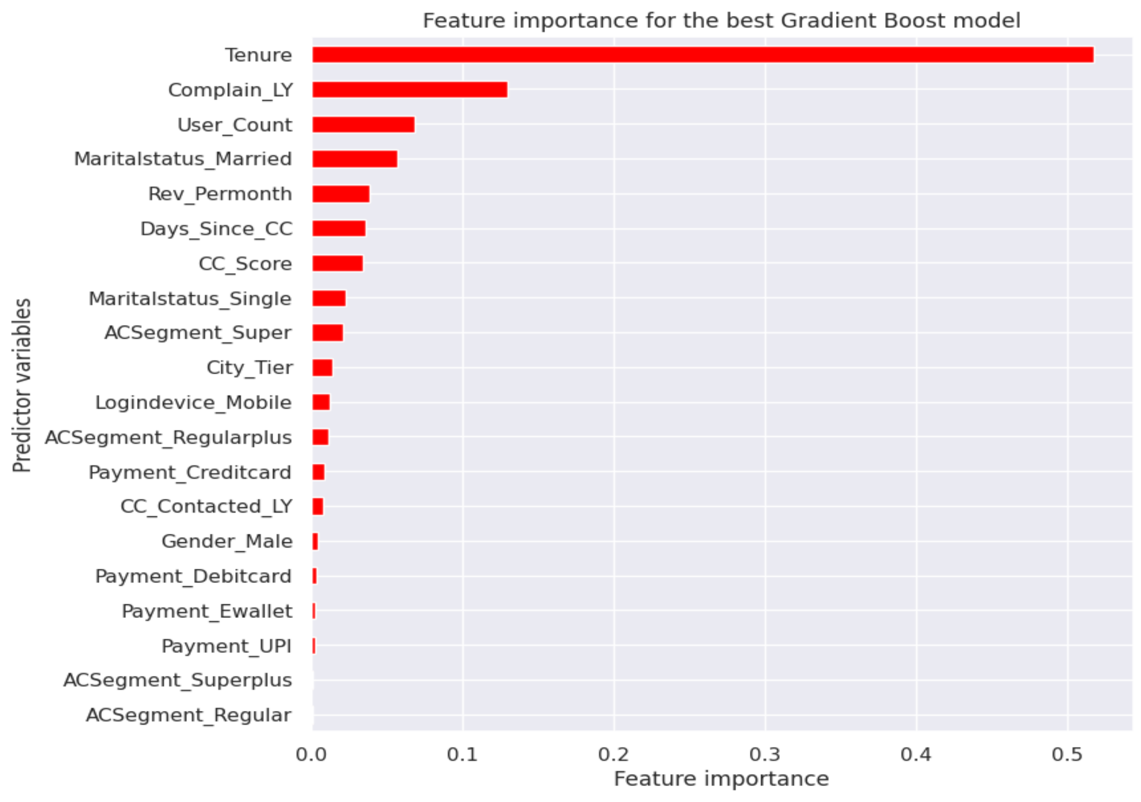


Fig.8.3 Feature Importance for the best Gradient Boost Model

10. Model Validation-

The given data was split into a train and test dataset in the ratio 70:30. The test dataset was held out and kept for validation purposes only.

- All models were trained only on the train dataset. The trained model was used to predict the train dataset target variable. The performance metrics such as Accuracy, F1-score, Precision, Recall, confusion matrix, ROC curve and AUC was observed and recorded on the train dataset.
- The trained model was then used to predict the target variable on the test dataset. All the above said performance metrics were observed and recorded for the test data performance as well.
- If train data seemed to be giving all 1's and the difference between train and test performances are not off by more than 10%, a validation of test scores was done by doing 5-fold and 10-fold cross validation on the entire dataset. The scores on this cross validation were compared to test dataset scores to ensure that model had not overfitted on train dataset.

10.1 Overall Model Building Comparison across parameters-

Training Dataset(70%) and Test Dataset(30%)

Training Dataset (70%)	**Test Dataset(30%)**

	Accuracy	**Precision**	**Recall**	**F1**	**AUC**	**Accuracy**	**Precision**	**Recall**	**F1**	**AUC**
LR	0.89	0.77	0.51	0.61	0.88	0.89	0.78	0.50	0.61	0.88
LR-CV	0.89	0.77	0.50	0.61	0.88	0.89	0.78	0.49	0.60	0.86
LR-SM	0.80	0.80	0.82	0.81	0.88	0.79	0.44	0.79	0.56	0.86
LDA	0.88	0.77	0.47	0.56	0.86	0.88	0.77	0.45	0.57	0.86
LDA-CV	0.88	0.77	0.47	0.59	0.87	0.88	0.77	0.45	0.59	0.85
LDA-SM	0.80	0.79	0.83	0.81	0.88	0.78	0.42	0.79	0.55	0.86
KNN	0.95	0.90	0.82	0.86	0.98	0.91	0.80	0.68	0.73	0.94
KNN-CV	1.0	1.0	1.0	1.0	1.0	0.95	0.90	0.82	0.86	0.97
KNN-SM	0.89	0.83	0.99	0.90	0.98	0.80	0.45	0.91	0.60	0.92

RF	1.0	1.0	0.98	1.0	0.82	0.97	0.97	0.87	0.92	0.80
RF - SM	1.0	1.0	0.97	1.0	0.84	0.97	0.97	0.88	0.92	0.82
NB	0.78	0.42	0.72	0.53	0.82	0.78	0.41	0.72	0.52	0.81
NB- SM	0.73	0.69	0.87	0.77	0.83	0.63	0.30	0.84	0.44	0.80
Bagging	0.99	1.0	0.98	0.99	1.0	0.96	0.96	0.88	0.92	0.99
Bagging - CV	0.99	1.0	1.0	1.0	1.0	0.96	0.93	0.86	0.89	0.98
Ada-Boost	0.89	0.74	0.61	0.67	0.91	0.89	0.75	0.60	0.67	0.990
Ada-Boost-CV	0.90	0.91	0.91	0.91	0.96	0.87	0.62	0.73	0.67	0.90
GB	0.92	0.83	0.67	0.74	0.94	0.91	0.81	0.61	0.70	0.92
GB - CV	0.93	0.95	0.93	0.94	0.98	0.90	0.73	0.72	0.72	0.91

Indicators/symbols for above tabular data-

- CV - indicates scores for models built on best params obtained from GridSearchCV with model name as prefix.
- SM - indicates scores for models built on a balanced dataset with model name as prefix.

10.2 Interpretation of the most optimum model-

- From the above tabular representation of all the scores for training and testing dataset across various models we can conclude that **the KNN model with GridSearchCV 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 the difference in accuracy for training and testing dataset is little on the higher side as compared to KNN.
- Other model's namely Naïve Bayes, LDA and RF worked well on training dataset but the accuracy came down when performed over testing dataset. Which indicates overfitting of data in that model.

10.3 Implication of final model on Business-

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

The End…