# **Capstone Project**

# Customer Churn Prediction (E-commerce or DTH company)

By:

Sharjil Shah

PGP – DSBA Jan22’ A

Great Learning

Contents

[1) Introduction of the business problem 5](#_Toc123948603)

[A) Defining problem statement 5](#_Toc123948604)

[B) Need of the study/project 5](#_Toc123948605)

[C) UNDERSTANDING HOW DATA WAS COLLECTED 5](#_Toc123948606)

[D) VISUAL INSPECTION OF DATA 6](#_Toc123948607)

[i) Dimension of the Dataset: 6](#_Toc123948608)

[ii) The variable details are as follows: 7](#_Toc123948609)

[2) EXPLORATORY DATA ANALYSIS 7](#_Toc123948610)

[A) Univariate analysis and bivariate analysis 7](#_Toc123948611)

[3) Data Cleaning and Pre-processing 13](#_Toc123948612)

[a) REMOVAL OF UNWANTED VARIABLES: 13](#_Toc123948613)

[b) Checked for Missing Values : 14](#_Toc123948614)

[c) Checking for Outliers 14](#_Toc123948615)

[d) Variable Transformation: 14](#_Toc123948616)

[d) Business insights from EDA 15](#_Toc123948617)

[a. Is the data unbalanced? If so, what can be done? Please explain in the context of the business 15](#_Toc123948618)

[b. Other Business insights: 16](#_Toc123948619)

[4) MODEL BUILDING: 16](#_Toc123948620)

[1.1 Build various models: 16](#_Toc123948621)

[1.2 Test your predictive model against the test set using various appropriate performance metrics (Model validation) 18](#_Toc123948622)

[1.3 Interpretation of the models 21](#_Toc123948623)

[1.3.1 Decsion tree classifier: 21](#_Toc123948624)

[1.3.2 Bagging Classifier 22](#_Toc123948625)

[1.3.3 Random Forest Classifier 23](#_Toc123948626)

[1.3.4 XGBoost Classifier 25](#_Toc123948627)

[5) Model Tuning and Business implication 27](#_Toc123948628)

[1.4 Other model tuning measures: 27](#_Toc123948629)

[*1.4.1* *Hyper-parameter tuning for Decision Tree model* 28](#_Toc123948630)

[1.4.2 Hyper- parameter tuning for Bagging Classifier model: 29](#_Toc123948631)

[2.1.4 Hyper- parameter tuning for Random Forest Classifier model: 30](#_Toc123948632)

[2.1.4 Hyper- parameter tuning for XGBoost Classifier model: 31](#_Toc123948633)

[2.2. Comparison of Train and Validation Performances 32](#_Toc123948634)

[2.2 Best Individual Models 33](#_Toc123948635)

[2.3 Stacking Classifier 34](#_Toc123948636)

[Stacking 1: 34](#_Toc123948637)

[Stacking 2: 35](#_Toc123948638)

[Stacking 3: 36](#_Toc123948639)

[Stacking 4: 37](#_Toc123948640)

[2.4 Comparison of Stacking Model Performances 38](#_Toc123948641)

[6) Interpretation of the most optimum model and its implication on the business 39](#_Toc123948642)

[2.5.1 Best Model – Performance 39](#_Toc123948643)

[Business Insights and Recommendations 40](#_Toc123948644)

# List of tables

[Table 1 - Table showing variable and description 7](#_Toc123948645)

[Table 1 - train test data split 17](#_Toc123948646)

[Table 2 - Cross Validation Performance 20](#_Toc123948647)

[Table 3 - Validation set performance 20](#_Toc123948648)

[Table 4 - Decision Tree Classifier Performance 22](#_Toc123948649)

[Table 5 -Bagging Classifier Performance 23](#_Toc123948650)

[Table 6 - Classification matrix for bagging classifier 23](#_Toc123948651)

[Table 7 - Random Forest model performance 24](#_Toc123948652)

[Table 8 - Confusion matrix for Random forest 24](#_Toc123948653)

[Table 9 - XGBoost model performance 26](#_Toc123948654)

[Table 10 - performance metrics for hypertuning DTC model 28](#_Toc123948655)

[Table 11 - Hypertuning performance for Bagging Classifier Model 29](#_Toc123948656)

[Table 12 - Performance metrix for stack 1 35](#_Toc123948657)

[Table 13 - Performance metrics for stack 2 36](#_Toc123948658)

[Table 14 - Performance metrics for Stack 3 37](#_Toc123948659)

# List of figures

[Figure 1 - Count plot for churn 9](#_Toc123948660)

[Figure 2 - Histogram plots for univariate analysis 13](#_Toc123948661)

[Figure 3 - Box plot for each variable in the data set for univariate analysis 14](#_Toc123948662)

[Figure 4 - Scatter plot for bivariate analysis. (Please refer Jupyter notebook) 15](#_Toc123948663)

[Figure 5 - output after treating missing values 16](#_Toc123948664)

[Figure 6- Variable Transformation 17](#_Toc123948665)

[Figure 7 - Output showing unbalanced data 17](#_Toc123948666)

[Figure 1 - Accurcy formula 19](#_Toc123948667)

[Figure 2 - Precision Formula 19](#_Toc123948668)

[Figure 3 - Recall Formula 20](#_Toc123948669)

[Figure 4 - F1 score formula 20](#_Toc123948670)

[Figure 5 - Algorithm comparision using Boxplot 22](#_Toc123948671)

[Figure 6 - Confusion matrix for decision tree classifier 24](#_Toc123948672)

[Figure 7 - Feature Importances 27](#_Toc123948673)

[Figure 8 - Confusion matrix for XGBoost 28](#_Toc123948674)

[Figure 9 - Feature importance bar plot 29](#_Toc123948675)

[Figure 10 -Output Hypertuning for Decision Tree Classifier model 30](#_Toc123948676)

[Figure 11 - Confusion matrix for hypertuned DCT model 30](#_Toc123948677)

[Figure 12 - Hyper-Parameter tuning for Bagging classifier 31](#_Toc123948678)

[Figure 13 - Confusion matrix for hypertuned Bagging classifier model 32](#_Toc123948679)

[Figure 14 - Tuning Random forest classifier model 32](#_Toc123948680)

[Figure 15 - performance metrics for tuned random forest classifier model 33](#_Toc123948681)

[Figure 16 - Confusion matrix for tuned random forest classifier model 33](#_Toc123948682)

[Figure 17 - Hyper-tuning XGBoost classifier 33](#_Toc123948683)

[Figure 18 - Hypertuned performance of XGBboost classifier 34](#_Toc123948684)

[Figure 19 - confusion Matrix for tuned XGB boost classifer 34](#_Toc123948685)

[Figure 20 -Training performance comparison 34](#_Toc123948686)

[Figure 21 - Best individual models train 35](#_Toc123948687)

[Figure 22 - Best model performance for validation 36](#_Toc123948688)

[Figure 23 - Stacking 1 36](#_Toc123948689)

[Figure 24 - Stacking 2 37](#_Toc123948690)

[Figure 25 - Confusion matrix for Stack 2 38](#_Toc123948691)

[Figure 26 - Stacking 3 38](#_Toc123948692)

[Figure 27 - Confusion matrix for Stack 3 39](#_Toc123948693)

[Figure 28 - Performance metrics and confusion matrix 40](#_Toc123948694)

# Introduction of the business problem

## Defining problem statement

An E Commerce company or DTH (you can choose either of these two domains) provider is facing a lot of competition in the current market and it has become a challenge to retain the existing customers in the current situation. Hence, the company wants to develop a model through which they can do churn prediction of the accounts and provide segmented offers to the potential churners. In this company, account churn is a major thing because 1 account can have multiple customers. Hence by losing one account the company might be losing more than one customer.

You have been assigned to develop a churn prediction model for this company and provide business recommendations on the campaign. Your campaign suggestion should be unique and be very clear on the campaign offer because your recommendation will go through the revenue assurance team. If they find that you are giving a lot of free (or subsidized) stuff thereby making a loss to the company; they are not going to approve your recommendation. Hence be very careful while providing campaign recommendation.

The objective of this case study is to understand the DTH industry and the challenges faced by the industry to protect their customers porting out to other providers due to high level competition. We need to help the company to predict customer churn based and help them increase the number of customers by preventing the customers from disconnecting the services basis of data provided.

## Need of the study/project

Customer churn greatly affects a company as they incur losses which impacts the Revenue of the company. It is vital to retain the customers on a long run for the company’s profitability and sustain the competition among other providers. If the Churn rate is high, then there is need of improvement in the services. There is a need to come forward with appropriate promotions and measures to resolve the customer’s problems, operational problems, management problems etc.

## UNDERSTANDING HOW DATA WAS COLLECTED

We have a data set of the online E-commerce company providing DTH (Direct-To-Home) services. The company is losing the users. We need to predict the churn status effectively and help the company to retain the customers as a part of marketing department in the company.

## VISUAL INSPECTION OF DATA

The Dataset provided in the case study is stored as “Customer+Churn+Data.xls”. It is imported in Jupyter Notebook using Pandas library and Python Language. The variables/columns of the data set are as follows in the table:

|  |  |
| --- | --- |
| **Variable** | **Description** |
| AccountID | account unique identifier |
| Churn | account churn flag (Target) |
| Tenure | Tenure of account |
| City\_Tier | Tier of primary customer's city |
| CC\_Contacted\_L12m | How many times all the customers of the account has contacted customer care in last 12months |
| Payment | Preferred Payment mode of the customers in the account |
| Gender | Gender of the primary customer of the account |
| Service\_Score | Satisfaction score given by customers of the account on service provided by company |
| Account\_user\_count | Number of customers tagged with this account |
| account\_segment | Account segmentation on the basis of spend |
| CC\_Agent\_Score | Satisfaction score given by customers of the account on customer care service provided by company |
| Marital\_Status | Marital status of the primary customer of the account |
| rev\_per\_month | Monthly average revenue generated by account in last 12 months |
| Complain\_l12m | Any complaints has been raised by account in last 12 months |
| rev\_growth\_yoy | revenue growth percentage of the account (last 12 months vs last 24 to 13 month) |
| coupon\_used\_l12m | How many times customers have used coupons to do the payment in last 12 months |
| Day\_Since\_CC\_connect | Number of days since no customers in the account has contacted the customer care |
| cashback\_l12m | Monthly average cashback generated by account in last 12 months |
| Login\_device | Preferred login device of the customers in the account |

Table 1 - Table showing variable and description

### Dimension of the Dataset:

The data set given has 11260 rows and 19 columns.

### The variable details are as follows:

1. *Churn* – This variable is a binary variable with output as 1 or 0. This will be the target variable.
2. Binary Variables namely “*Gender*”, “*Complain\_ly*” have only two outputs.
3. Numerical variables namely “Tenure”, “*CC\_Contacted\_LY*”, “*coupon\_used\_for\_payment*”, “*Day\_Since\_CC\_connect*” and “*cashback*” are continuous variables.
4. Variables “*City\_Tier*”, “*Payment*”, "*Service\_Score*",” *account\_segment*", "*CC\_Agent\_Score*", "*Marital\_Status*" and "*Login\_device*" are categorical.
5. We will convert the binary and categorical variables and *Account\_user\_count*, *rev\_per\_month* and cashback to numerical variable

# EXPLORATORY DATA ANALYSIS

## Univariate analysis and bivariate analysis

#### Churn:

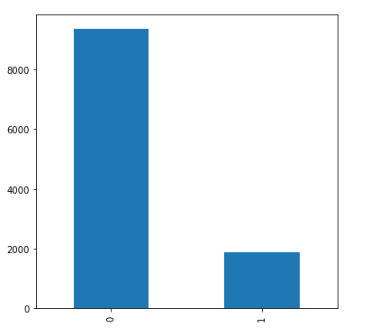


Figure 1 - Count plot for churn

This is a binary variable and our target variable. This is a binary variable and our target variable. The data is unbalanced as 16.7% churned customers and remaining 83.3% non-churned customers.

#### Tenure:

This is a type of numerical variable which is discrete in nature. This variable tells us about the number of months customer is using the service. The minimum tenure of customers were 0-5 months whereas average being 11 years. Outliers are present in column variable as the data is spread over 60 months.

#### City Tier:

This is a categorical variable with three values 1, 2, 3. 58% of total customers belong to Tier 1 cities. 22.2% in tier 3, and the rest of the customers in the Tier 2. We can see the comparison in the pairplot between tier and the target variable Churn. Most of the customers those are churned belong to City Teir 1.

The number of customer who are churned are far lesser than the number of customers who are retained.

#### Customer Care contacted in the last year:

This variable is numerical but discrete in nature. It provides the information about how many times the user contacted customer service team in the previous year. Outliers are present in this variable.

#### Payment:

Categorical variable. It has 5 discrete values – Cash on delivery, Delivery, Credit card, E-Wallet, Debit card, and UPI. Approximately 40% of the customers have preferred Debit card as payment mode. 9.7% people have preferred E-Wallet. As per the analysis, most of the churned customers have opted debit card as payment method.

#### Gender:

This variable is a binary variable with two values i.e. Male and Female. Originally in the dataset had “F” and “M” letters which we needed to combine with “Female” and “Male” respectively. This is because the system was categorizing the data points in 4 variables whereas the genders were in only two categories. Approximately 57.7% are males and remaining are females. As per the Bivariate analysis, we have compared the Gender with our target variable, Churn. The number of Churned customers is far less than the number of retained customers.

#### Service Score:

Categorical variable. It has 5 categorical values. Service code 3 is given to approximately 44% of the customers. 26% as 2 and 22.2% as 4. More number of churned customers are having service code 3.

#### Account user:

This variable represents the number of users per account. It is discrete numerical variable. The max number of users per account is 4. We have compared Account user variable with the target variable Churn. The number of retained customers is very high when we compare with those who churned.

#### Account segment:

This variable is a categorical variable with five discrete values. In the dataset there were 7 categories because of the anomalies. “Regular +” and “Super +” categories were changed to “Regular Plus” and “Super Plus” respectively to form 5 categories.

The total customers approximately 32.9% of the customers have “Regular Plus” account category followed by 32% of the customers who have “Super” Category and 13.3% as “HNI” category.

Most of the Churned customers have Regular Plus segment.

#### Customer Care Agent Score:

Categorical variable with five discrete values. Total customers approximately 27.5% of the customers have given an agent score of ‘3’ followed by 19.5% as ‘2’ and 17.76% as ‘5’. Majority of the customers have given the score of 3 which means, there are neither unhappy nor happy with the customer care service agent.

#### Marital Status:

This is categorical variable with 3 discrete values. Approximately 48.8% of the customers are married, 28.4% are singles and remaining divorced. Majority of churned customers have marital status as single.

#### Revenue per month:

It describes the average income generated by an account in a month. It is continuous or numerical variable. For most of the accounts the income generated per month is between 0 and 10. The average is 6. Outliers are present since 2% of the data is spread over 120. Most of the churned accounts had revenue per month between 0 and 5.

#### Complain last year:

This is a categorical variable. The customer who complained is represented by 1. The customer who has not complained is represented by 0. 66.6% of the customers have not made complaints. 33% of the customers have made complaints. Evidently, more customers who are churned have complained.

Revenue growth year on year:

Numerical variable. It represents revenue growth in percentage. Revenue growth per month for most of the customers is between 12 and 14. Average is 16. Churned customers had the revenue growth between 12 and 14.

#### Coupon used for payment:

This is a type of numerical variable which is discrete in nature. It tells us the number of coupons used for payment by customers. For most of the customers the number of coupons used were between 0-2. Churned customers have used around 0-2 coupons for payment with certain outliers present. The number of Churned customers is far less than the number of retained customers.

#### Day since customer connect:

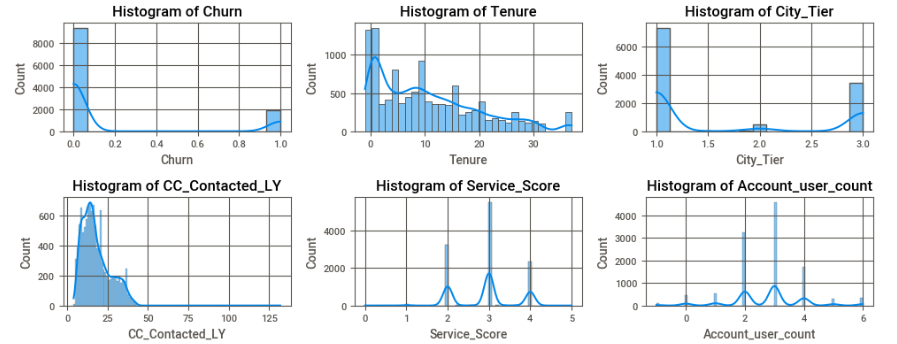
This is a type of numerical variable which is discrete in nature. It tells us the day since the customers last connected with the customer care. For majority of customers the days since last Customer care connect are between 0-5 days. Majority of “Churned” customers have had the customer care connect in the last 0-2 days. The customers that are retained with the company have had the last Customer care connect in the last 0-12 days with outliers present.

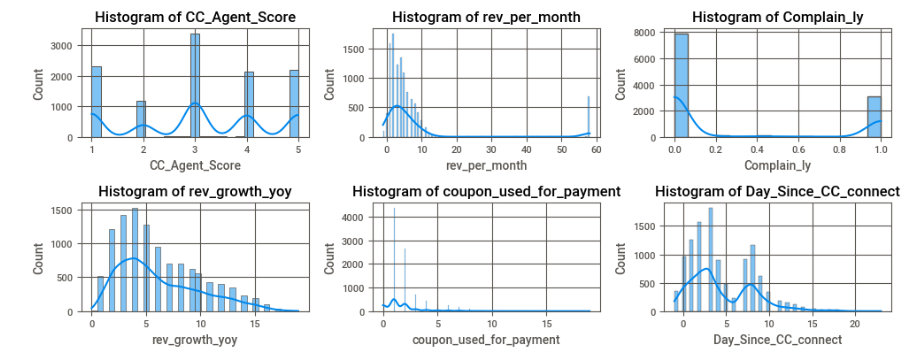
#### Cashback amount:

This represents the amount earned by the customers for the payments they made. Continuous variable. Most of customers the cashback amount earned range between 100-200 whereas average being 170. “Churned” customers have earned cashback amount between 140-160 with certain outliers present. The customers that are retained with the company have earned cashback amount between 120-320 with outliers present.

#### Login Device:

A categorical variable with three values Mobile phones, Computer and Both. The 62.16% of the customers use mobile phones as their preferred login device. 26.6% of the customers use computers and remaining use both the devices. We have compared the Login device with our target variable. Most of churned customers had their preferred login device as mobile phones.





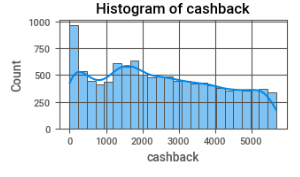
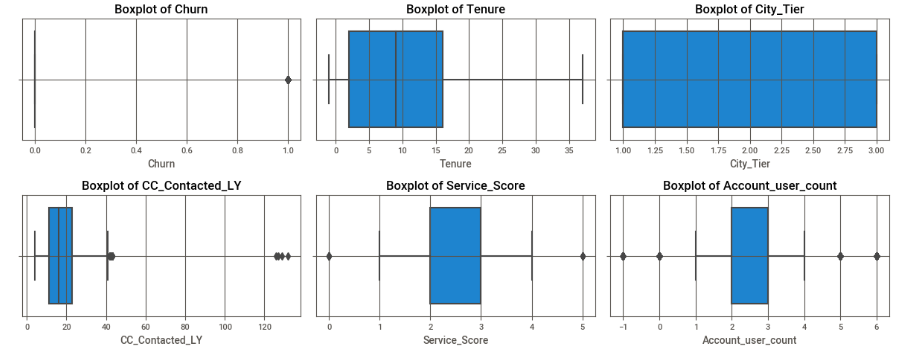
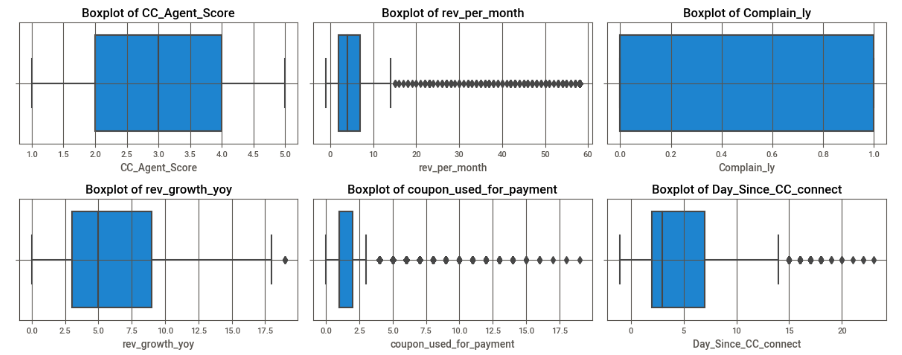


Figure 2 - Histogram plots for univariate analysis





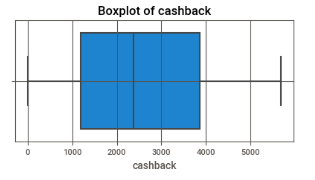


Figure 3 - Box plot for each variable in the data set for univariate analysis

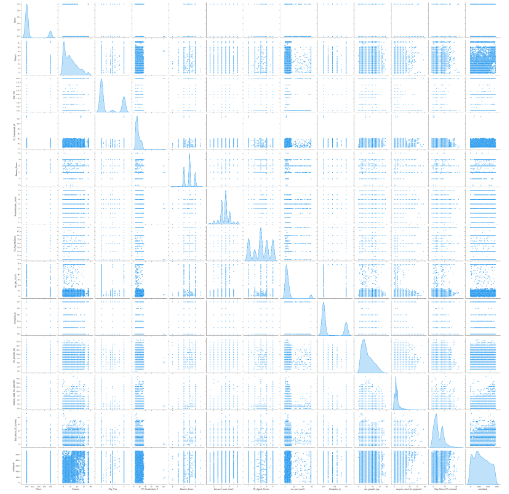


Figure 4 - Scatter plot for bivariate analysis. (Please refer Jupyter notebook)

# Data Cleaning and Pre-processing

### a) REMOVAL OF UNWANTED VARIABLES:

‘CustomerID’ column is removed. This is unwanted variable because it is just a serial number and the data is already in the table. This doesn’t provide valuable insight for the analysis.

### b) Checked for Missing Values :

We found many missing values in the data set. It’s not a good idea that we remove the missing values. These missing data could be influencing the accuracy of the model that we are going to build. We have less number of churn customers compared with retained so we impute the missing values. We will further treat them by replacing the missing values with “No\_info” in the variables i.e. “Tenure”, CC\_Contacted\_LY”, “Account\_user\_count”, “rev\_per\_month”, “Complain\_LY”, rev\_growth\_yoy”, “coupon\_used\_for\_payment”, “day\_since\_CC\_Connect”, “cashback”. Using KNN imputer from Scikit (sklearn) library, we have imputed the missing values by evaluating the Euclidian distances between the nearest neighbors.

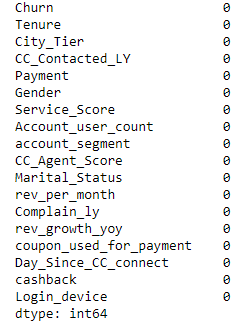


Figure 5 - output after treating missing values

### c) Checking for Outliers

We have found all the variables having outliers. It is necessary to be treated. We have grouped category variables and numerical variables. We have imputed categorical variables within the range of the the quartiles.

### d) Variable Transformation:

While performing EDA we came across three variables namely Gender, account segment and login device which had some extra categories. By the virtue of similar names, we have combined the categories under similar headings.

Gender: “F” – “Female” and “M” – “Male”

Account segment: “Regular +” – “Regular Plus” and “Super +” – “Super Plus”

Login Device: “&&&&” – “both”

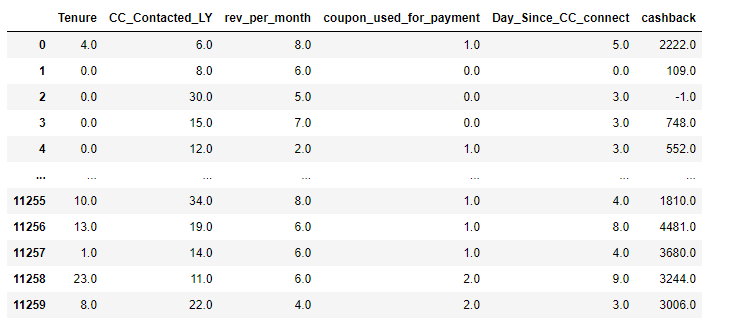


Figure 6- Variable Transformation

### Business insights from EDA

### Is the data unbalanced? If so, what can be done? Please explain in the context of the business

The data is highly unbalanced. There are 83.16% retained customers in the dataset and 16.83% churned customers only. SMOTE (synthetic minority oversampling technique) is one of the most commonly used oversampling methods to solve the imbalance problem.

It aims to balance class distribution by randomly increasing minority class examples by replicating them.

SMOTE synthesizes new minority instances between existing minority instances. It generates the virtual training records by linear interpolation for the minority class. These synthetic training records are generated by randomly selecting one or more of the k-nearest neighbors for each example in the minority class. After the oversampling process, the data is reconstructed and several classification models can be applied for the processed data.

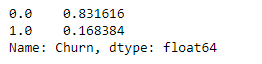


Figure 7 - Output showing unbalanced data

### Other Business insights:

The customers looks like monthly subscribers. There should’ve been information about their recent relocation. The customers those who disconnected the services, may have connected back to the same provider in the new city.

# MODEL BUILDING:

## Build various models:

#### Splitting the Data into Train and Test Dataset (75:25)

The first step of any Machine Learning Model Building is splitting the dataset into training and testing datasets. This is majorly done to evaluate the performance of an algorithm. This is done by dividing the dataset into two subsets where the training dataset is used to fit the model whereas the testing dataset is used to test the model accuracy and fit. We will split the dataset in a ratio of 75:25 by using the library sklearn.model\_selection in Python.

Shape of Training set : (6756, 17)

Shape of Validation set : (2252, 17)

Shape of Testing set : (2252, 17)

Percentage of classes in training set :

0.0 0.831557

1.0 0.168443

Name: Churn, dtype: float64

Percentage of classes in validation set :

0.0 0.831705

1.0 0.168295

Name: Churn, dtype: float64

Percentage of classes in test set :

0.0 0.831705

1.0 0.168295

Name: Churn, dtype: float64

Table 1 - train test data split

#### Model can make wrong predictions such as:

* Predicting a customer will churn but in reality the customer will not.
* Predicting a customer will not quit the service but in reality the customer will churn

#### Prediction of concern:

The second prediction is our major concern as customers renouncing the services would lead to loss and our aim is to build a prediction model to minimize the churn

#### Minimizing false negatives:

**Recall** score should be maximized. Greater the Recall score, higher the chances of predicting the customers who may churn

#### User-defined functions to evaluate and plot metric scores

We are defining a function to compute different metrics to check performance of classification models built using sklearn library. For reference please check the Jupyter notebook.

Function to compute different metrics to check classification model performance

model: classifier

predictors: independent variables

target: dependent variable

#### Creating a dataframe of metrics

**Accuracy**: The number of correct predictions over all predictions.

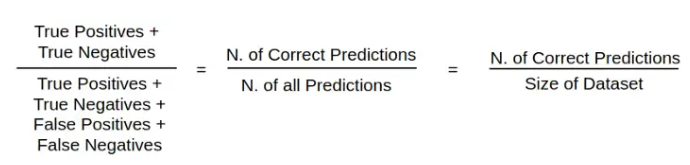


Figure 1 - Accurcy formula

**Precision**: Precision is a measure of how many of the positive predictions made are correct (true positives).

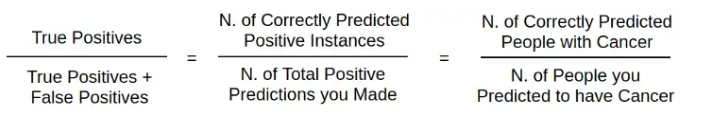


Figure 2 - Precision Formula

**Recall / Sensitivity:** Recall is a measure of how many of the positive cases the classifier correctly predicted, over all the positive cases in the data.

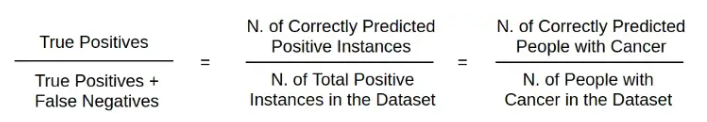


Figure 3 - Recall Formula

**F1 Score**: F1-Score is a measure combining both precision and recall. It is generally described as the harmonic mean of the two. Harmonic mean is just another way to calculate an “average” of values, generally described as more suitable for ratios (such as precision and recall) than the traditional arithmetic mean.

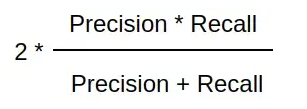


Figure 4 - F1 score formula

Very small precision or recall will result in lower overall score. Thus it helps balance the two metrics. If you choose your positive class as the one with fewer samples, F1-score can help balance the metric across positive/negative samples.

**Target variable**: In the given data set, ‘Churn’ is the target variable

## Test your predictive model against the test set using various appropriate performance metrics (Model validation)

We created functions to compute different metrics to check classification model performance and to plot the confusion matrix with percentages.

model: classifier

predictors: independent variables

target: dependent variable

In order to perform cross-validation, the following steps are typically taken:

* + Split the dataset into training data and test data
  + The parameters will undergo a Cross-Validation test to see what the best parameters to select are.
  + These parameters will then be implemented into the model for retraining
  + Final evaluation will occur and this will depend if the cycle has to go again, depending on the accuracy and the level of generalization that the model performs.

K-fold Cross-Validation is when the dataset is split into a K number of folds and is used to evaluate the model's ability when given new data. K refers to the number of groups the data sample is split into.

When using K-fold cross validation, all parts of the data will be able to be used as part of the testing data. This way, all of our data from our small dataset can be used for both training and testing, allowing us to better evaluate the performance of our model.

We appended the models to the list and looped through all the models to get the mean cross validated score

Models we used:

* Logistic Regression
* Decision Tree Classifier
* Bagging Classifier
* Random Forest Classifier
* Ada Boost Classifier
* Gradient Boosting Classifier
* XGB Classifier

#### Cross-Validation Performance :

|  |  |
| --- | --- |
| LR | 37.43488678 |
| Dtree | 76.71535667 |
| Bagging | 72.23587603 |
| RandomForest | 76.36447948 |
| Adaboost | 55.62601438 |
| GBM | 57.82440683 |
| XGBoost | 79.96676714 |

Table 2 - Cross Validation Performance

#### Validation set Performance:

|  |  |
| --- | --- |
| LR | 42.21635884 |
| Dtree | 82.58575198 |
| Bagging | 79.94722955 |
| RandomForest | 82.58575198 |
| Adaboost | 61.21372032 |
| GBM | 61.47757256 |
| XGBoost | 84.69656992 |

Table 3 - Validation set performance

#### Algorithm Comparison:

Plotting boxplots for CV scores of models defined above.

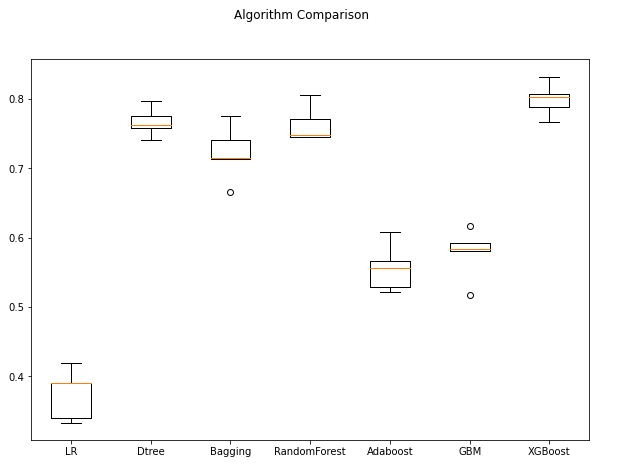


Figure 5 - Algorithm comparision using Boxplot

Inference:

* All the above models are consistent without any outliers
* Highest validation set Recall achieved is 88 by XG Boost.
* XG Boost is giving the highest mean cross-validated Recall followed by Random Forest, Decision Tree and Bagging classifiers
* We built the models, hyper-tune and stack them to get much better results.

## Interpretation of the models

### Decsion tree classifier:

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree.

#### Calculating different metrics:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Training Performance | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 1 | 1 | 1 | 1 |
|  | | | | |
| Validation Performance | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.936945 | 0.825858 | 0.804627 | 0.815104 |

Table 4 - Decision Tree Classifier Performance

#### Confusion matrix:

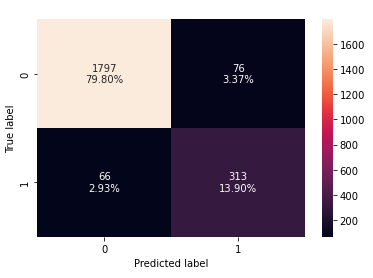


Figure 6 - Confusion matrix for decision tree classifier

The model is overfitting the train data with default parameters.

### Bagging Classifier

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random sets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.

#### Calculating different metrics:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Training Performance | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.996004 | 0.979789 | 0.996425 | 0.988037 |
|  |  |  |  |  |
| Validation Performance | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.954263 | 0.799472 | 0.918182 | 0.854725 |

Table 5 -Bagging Classifier Performance

#### Confusion matrix:

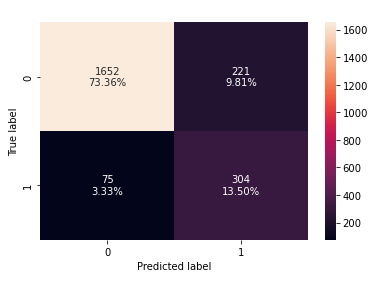


Table 6 - Classification matrix for bagging classifier

### Random Forest Classifier

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

#### Calculating different metrics:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Training Performance | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 1 | 1 | 1 | 1 |
|  |  |  |  |  |
| Validation Performance | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.963588 | 0.825858 | 0.951368 | 0.884181 |

Table 7 - Random Forest model performance

#### Confusion matrix:

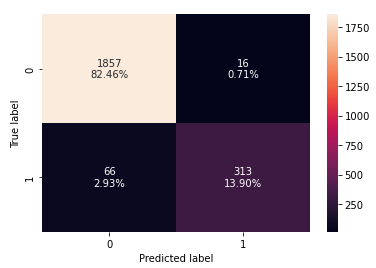


Table 8 - Confusion matrix for Random forest

#### Important features:

Imp

|  |  |
| --- | --- |
|  | Importance |
| Tenure | 0.25455 |
| cashback | 0.087726 |
| CC\_Contacted\_LY | 0.0703 |
| Day\_Since\_CC\_connect | 0.069844 |
| Complain\_ly | 0.065625 |
| rev\_growth\_yoy | 0.060278 |
| rev\_per\_month | 0.054532 |
| CC\_Agent\_Score | 0.054427 |
| Payment | 0.04558 |
| account\_segment | 0.03956 |
| Marital\_Status | 0.038855 |
| Account\_user\_count | 0.037935 |
| coupon\_used\_for\_payment | 0.028591 |
| City\_Tier | 0.026126 |
| Login\_device | 0.025066 |
| Service\_Score | 0.021237 |
| Gender | 0.019767 |

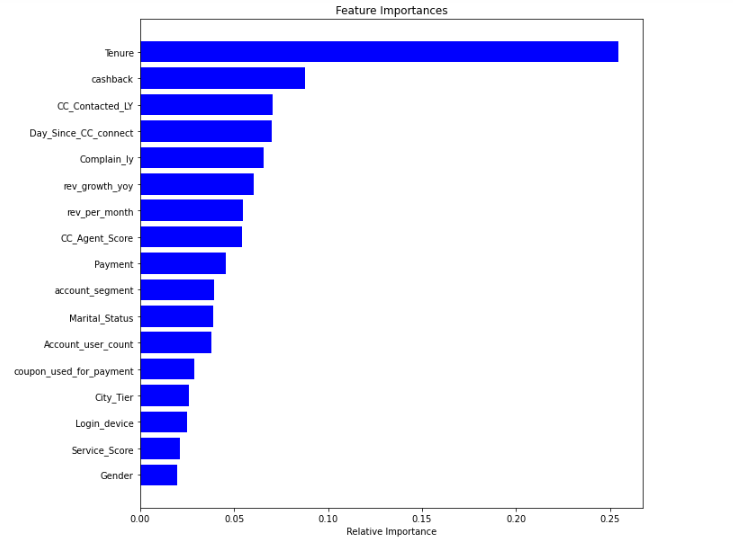


Figure 7 - Feature Importances

* + Tenure is the most important feature.

### XGBoost Classifier

The XGBoost stands for eXtreme Gradient Boosting, which is a boosting algorithm based on gradient boosted decision trees algorithm. XGBoost applies a better regularization technique to reduce overfitting, and it is one of the differences from the gradient boosting. The xgboost.XGBClassifier is a scikit-learn API compatible class for classification.

#### Calculating different metrics:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Training Performance | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.999704 | 0.998243 | 1 | 0.99912 |
|  |  |  |  |  |
| Validation Performance | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.963144 | 0.846966 | 0.927746 | 0.885517 |

Table 9 - XGBoost model performance

#### Confusion matrix:

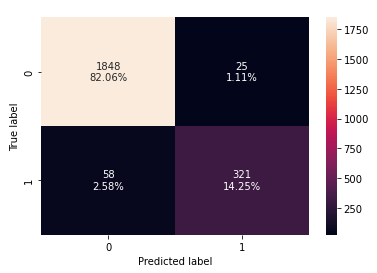


Figure 8 - Confusion matrix for XGBoost

#### Feature Importance:

|  |  |
| --- | --- |
|  | Importance |
| Tenure | 0.183042 |
| Complain\_ly | 0.136423 |
| Marital\_Status | 0.069657 |
| account\_segment | 0.067497 |
| City\_Tier | 0.062695 |
| Day\_Since\_CC\_connect | 0.052955 |
| CC\_Agent\_Score | 0.052544 |
| Gender | 0.04829 |
| rev\_per\_month | 0.044549 |
| Payment | 0.043945 |
| CC\_Contacted\_LY | 0.039995 |
| Account\_user\_count | 0.039992 |
| Login\_device | 0.038394 |
| coupon\_used\_for\_payment | 0.033213 |
| cashback | 0.032431 |
| rev\_growth\_yoy | 0.03206 |
| Service\_Score | 0.022319 |

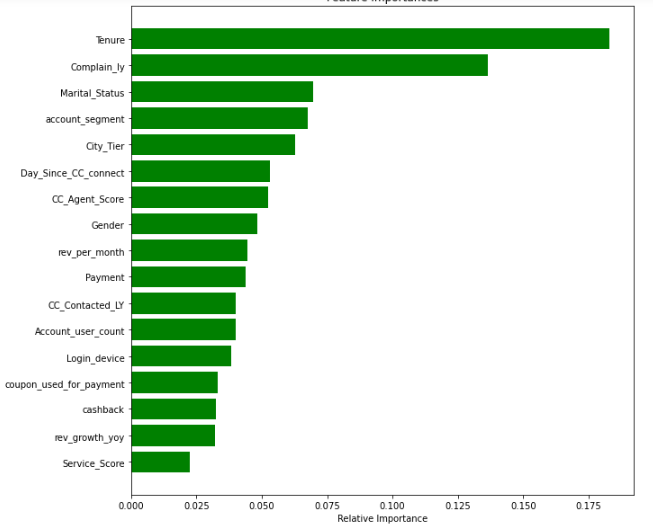


Figure 9 - Feature importance bar plot

Tenure is given the highest feature importance.

# Model Tuning and Business implication

### Other model tuning measures:

The models – Decision Tree, Random Forest, XGBoost, were overfitting the train data with default parameters. We hyper tune to reduce the overfitting.

#### Hyper-parameter tuning for Decision Tree model:

Using GridSearchCV selecting recall scoring, we find the best parameters. W set the classifier to the best combination of parameters. The output from the jupyter notebook is presented as below.

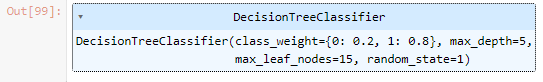


Figure 10 -Output Hypertuning for Decision Tree Classifier model

#### Calculating different metrics:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Training Performance | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.829929 | 0.792619 | 0.49697 | 0.610904 |
|  |  |  |  |  |
| Validation Performance | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.840586 | 0.796834 | 0.517123 | 0.627207 |

Table 10 - performance metrics for hypertuning DTC model

#### Confusion Matrix:

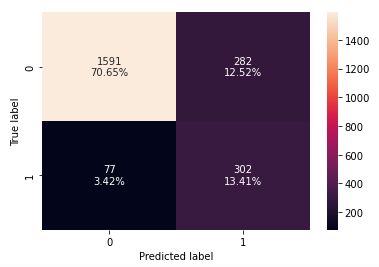


Figure 11 - Confusion matrix for hypertuned DCT model

The model is generalizing well with tuned parameters but with low metric scores.

### Hyper- parameter tuning for Bagging Classifier model:

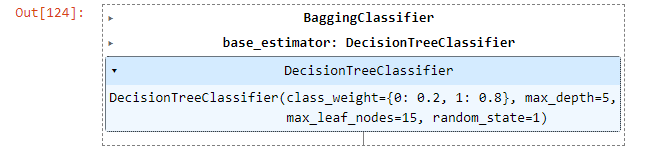
Since the model is generalizing well, but precision is too low. We will try to improve the scores by tuning this model. The most important parameter for bagged decision trees is the number of trees (n\_estimators). Grid searching the key hyper-parameters for BaggingClassifier.

Figure 12 - Hyper-Parameter tuning for Bagging classifier

#### Calculating different metrics:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Training Performance | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 1 | 1 | 1 | 1 |
|  |  |  |  |  |
| Validation Performance | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.962256 | 0.831135 | 0.9375 | 0.881119 |

Table 11 - Hypertuning performance for Bagging Classifier Model

#### Confusion matrix:

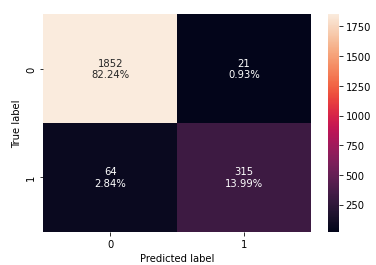


Figure 13 - Confusion matrix for hypertuned Bagging classifier model

Model is overfit on the train data. Precision is much higher than Recall

### Hyper- parameter tuning for Random Forest Classifier model:

We try tuning the model by adding weights. Tenure is given the highest feature importance. Most of the newly created features have been given importance in model building with user\_count\_ss with much higher significance. We run the grid search and set the best combination of parameters. Below is the output from Jupyter notebook is shown as below.

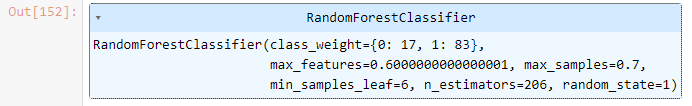


Figure 14 - Tuning Random forest classifier model

#### Calculating different metrics:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Training Performance | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.957667 | 0.963972 | 0.817437 | 0.884677 |
|  |  |  |  |  |
| Validation Performance | | | |  |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.928064 | 0.852243 | 0.752914 | 0.799505 |

Figure 15 - performance metrics for tuned random forest classifier model

#### Confusion matrix:

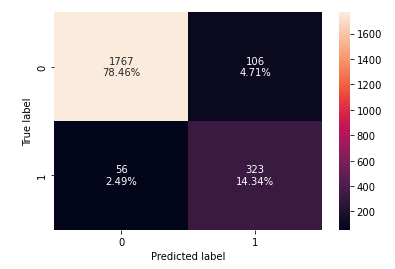


Figure 16 - Confusion matrix for tuned random forest classifier model

We observe better performance on the train data. There is no generalized performance on the validation data.

### Hyper- parameter tuning for XGBoost Classifier model:

The model was Model overfit on the train data and Precision much higher than Recall. We chose to do tuning on this model. After choosing the best parameter and running grid search, we fit the model with the best estimated for tuning. Below is the output from the Jupyter notebook.

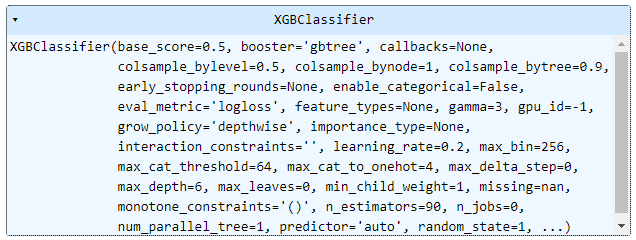


Figure 17 - Hyper-tuning XGBoost classifier

#### Calculation of different metrices:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Training Performance | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.98357 | 0.998243 | 0.91245 | 0.95342 |
|  |  |  |  |  |
| Validation Performance | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.946714 | 0.907652 | 0.801865 | 0.851485 |

Figure 18 - Hypertuned performance of XGBboost classifier

#### Confusion matrix:

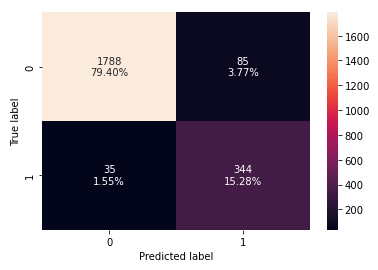


Figure 19 - confusion Matrix for tuned XGB boost classifer

Tenure is given the highest importance. Each newly created feature has some significance in model building

### 2.2. Comparison of Train and Validation Performances

#### Training performance comparison:

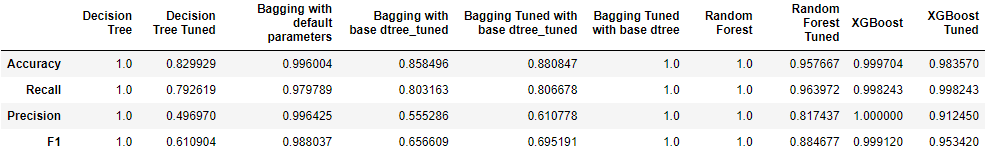
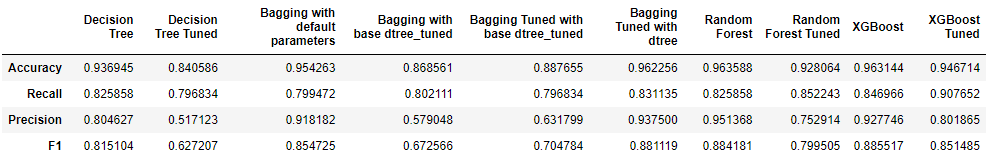


Figure 20 -Training performance comparison

#### Validation performance comparison:



* Decision tree, Bagging and Random forest models with default parameters are overfit
* Tuned decision tree is giving a generalized performance
* Tuned bagging classifier with dtree\_tuned as base estimator is generalizing well
* Tuned random forest is also overfit
* Tuned xgboost model is giving much higher recall than the model with default parameters

## Best Individual Models

#### Training performance comparison :

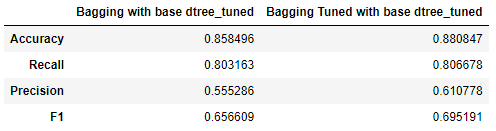


Figure 21 - Best individual models train

#### Validation performance comparison:

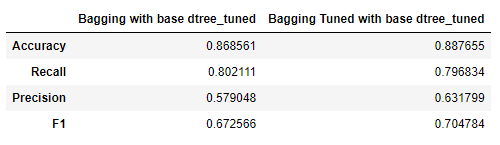


Figure 22 - Best model performance for validation

* Both these models are generalizing well on the train and validation sets
* Highest Recall achieved is 80 by Bagging Classifier with tuned decision tree as base estimator

## Stacking Classifier

Will build stacking models with combinations of above models with tuned xgboost model as the final estimator to achieve higher Recall with good Precision scores. Compare the stacking model performances to identify the best model

### Stacking 1:

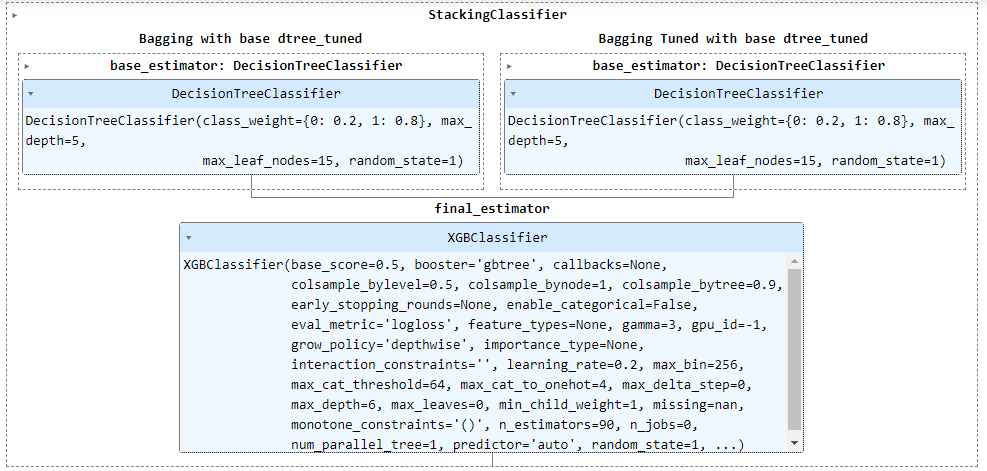


Figure 23 - Stacking 1

#### Performance metrices:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Training | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.847543 | 0.801406 | 0.531469 | 0.639103 |
|  |  |  |  |  |
| Validation | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.85746 | 0.802111 | 0.552727 | 0.654467 |

Table 12 - Performance metrix for stack 1

Model is generalizing well on both train and validation sets.

The Precision is too low.

### Stacking 2:

Building a stacking classifier with the best individual models and bagging classifier with default parameters, as it gave higher Recall and Precision scores.

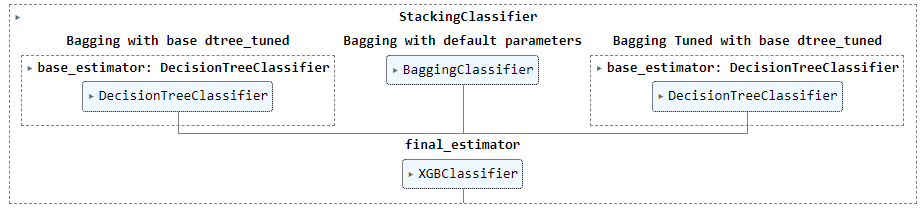


Figure 24 - Stacking 2

#### Performance metrices:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Training | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.987271 | 0.997364 | 0.931856 | 0.963497 |
|  |  |  |  |  |
| Validation | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.939609 | 0.926121 | 0.764706 | 0.837709 |

Table 13 - Performance metrics for stack 2

#### Confusion metrix:

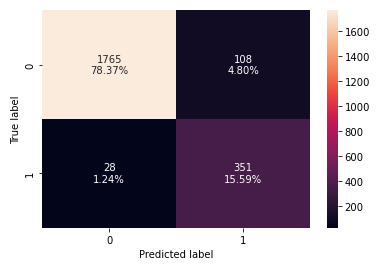


Figure 25 - Confusion matrix for Stack 2

* Model overfit on train data
* Highest Recall on validation so far
* Precision has improved comparatively

### Stacking 3:

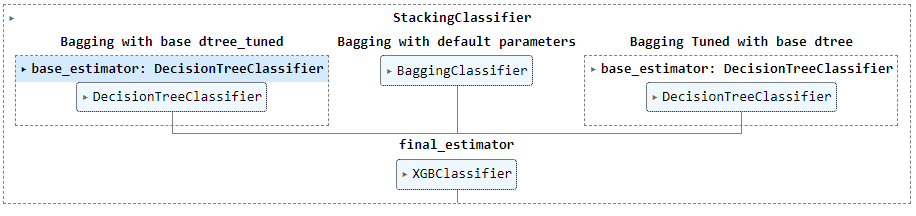


Figure 26 - Stacking 3

#### Performance metrics:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Training | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.99852 | 1 | 0.991289 | 0.995626 |
|  |  |  |  |  |
| Validation | | | | |
|  | Accuracy | Recall | Precision | F1 |
| 0 | 0.949822 | 0.939314 | 0.798206 | 0.86303 |

Table 14 - Performance metrics for Stack 3

#### Confusion matrix:

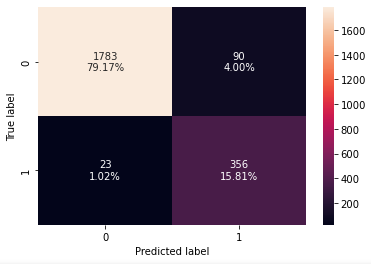


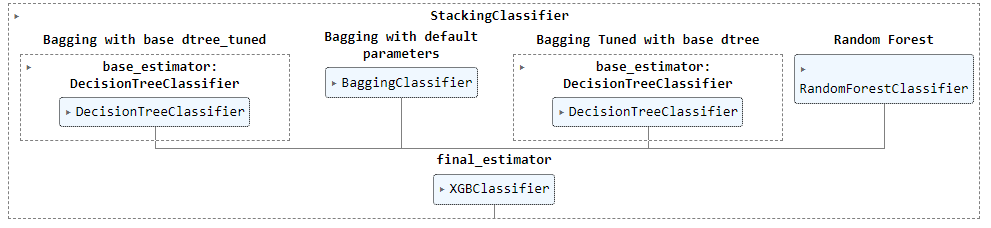
Figure 27 - Confusion matrix for Stack 3

Highest Recall score so far

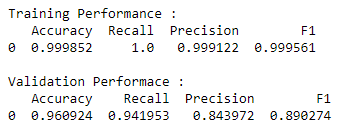
However, model is much overfit on Precision comparatively

### Stacking 4:

Stacking the above model with random forest classifier with base estimators in order to get a better Precision score.



Performance metrics and confusion matrix:



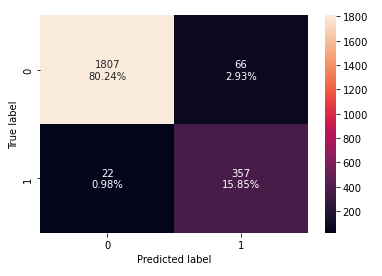
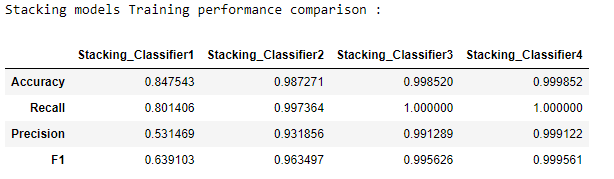
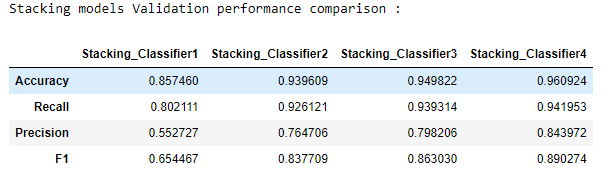


Figure 28 - Performance metrics and confusion matrix

## Comparison of Stacking Model Performances



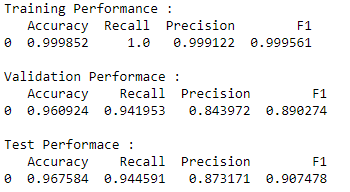


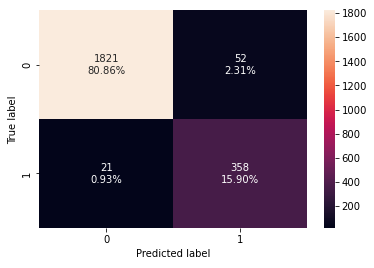
* Stacking Classifier 1 is generalizing well but, very low metric scores
* Stacking Classifier 2 gives a good Recall but the Precision is very low comparatively
* Stacking Classifier 3 gives the highest Recall however, it is much overfit on Precision
* Stacking Classifier 4 is a little overfit on the train set however, the Recall and Precision are really good
* **Hence, I conclude Stacking Classifier 4 as my best model**

# Interpretation of the most optimum model and its implication on the business

### 2.5.1 Best Model – Performance

Calculating different metrics for the best model across Train, Validation and Test sets.





* Although this model is a little overfit on the train set, it is giving a generalized performance on validation and test sets
* As the metric scores on the unseen data (Test set) are still higher than the validation set, it is safe to use this model

### Business Insights and Recommendations

1. The business can use this model to identify customers who may churn.
2. Top five features that drive the churn are Tenure, Complain\_ly, Marital\_Status\_Single, account\_segment and Single\_M
3. Customer care team must make additional efforts to solve customer complaints at the earliest as majority of customers who raised a complaint in the past year has churned. Follow-up calls are recommended.
4. Marketing team can target Single/M customers with special discounts on paid channels and/or movies to customers from Regular Plus, Super and Super Plus segments
5. Provide targeted offers to Female customers who prefer E-wallet/Mobile, from the Regular account segments.
6. Also provide exclusive family offers for Married customers from HNI segment as churn rate is higher among them
7. Business may consider increasing the cashback to Regular Plus customers and Debit card payment in order to reduce the churn
8. Exit interview can be conducted to get feedback from outgoing customers and work towards the betterment of the services provided
9. Business may provide introductory offers to attract new customers and exclusive offers to existing new customers

\*\*\*\*\*\*\*\*