Capstone Project

Customer Churn

Hritika Vaishnav

14 April 2024

Content of Report

Introduction of the Business Problem

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.

Defining Problem Objective:

The objective is to construct a Churn prediction model for an E-Commerce company (or DTH provider) to forecast and mitigate customer churn. Customer churn denotes the termination of services or subscriptions by customers. The task involves identifying churn-contributing factors and developing a predictive model to anticipate which accounts are likely to churn in the future.

Need of the Study/Project:

1. **Customer Retention:** It's imperative for business growth and profitability to retain customers. Understanding churn patterns aids in formulating strategies to retain valuable customers.
2. **Revenue Stability:** By reducing churn rates, the company can maintain a consistent revenue stream. Acquiring new customers typically incurs higher costs than retaining existing ones.
3. **Competitive Edge:** In a competitive market, customer satisfaction and loyalty differentiate a company from its competitors.
4. **Resource Optimization:** Predictive modeling allows for targeted interventions, optimizing resource allocation toward retaining high-risk customers.

Understanding Business/Social Opportunity:

1. **Personalized Offerings:** Analyzing customer behavior enables tailored marketing campaigns, enhancing satisfaction and loyalty.
2. **Enhanced Customer Experience:** Anticipating churn enables proactive customer service, resolving issues before they escalate and lead to dissatisfaction.
3. **Market Insights:** Analyzing churn patterns provides valuable insights into market trends, preferences, and areas for improvement.

By addressing these aspects, the company can develop effective strategies to mitigate churn, enhance customer satisfaction, and maintain a competitive position in the market.

EDA and Business Implications

Import all the necessary and load our data set, Customer Churn Data1CompData-1.xlsx and use the head() function to view the Top 5 data and the tail() function to view the bottom 5 data. Using the shape function, we can determine that there are 11260 rows and 19 column and size of dataset is 213940. Find out the characteristics of the column using the info() method. The datatypes for the float64(5), int64(2), and object(12) are present and 2676 null values are present in the dataset.

Sample of dataset: The provided dataset consists of E-Commerce Company (or DTH provider) to forecast and mitigate customer churn. Each entry includes details, such as 19 columns of features. Additionally, there is a column indicating whether a customer churn or not.

	AccountID	Churn	Tenure	City_Tier	CC_Contacted_LY	Payment	Gender	Service_Score	Account_u
0	20000	1	4	3.0	6.0	Debit Card	Female	3.0	
1	20001	1	0	1.0	8.0	UPI	Male	3.0	
2	20002	1	0	1.0	30.0	Debit Card	Male	2.0	
3	20003	1	0	3.0	15.0	Debit Card	Male	2.0	
4	20004	1	0	1.0	12.0	Credit Card	Male	2.0	

Table 1 – glimpse of data-frame head with top 5 rows

Understanding of attributes (variable info, renaming if required) Checking data types of all columns: There are total 11260 rows and 19 columns in the dataset. Out of 19, 12 columns are of object type, 2 columns are of integer type and rest 5 columns are of float data type.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11260 entries, 0 to 11259
Data columns (total 19 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   AccountID               11260 non-null  int64
 1   Churn                   11260 non-null  int64
 2   Tenure                  11158 non-null  object
 3   City_Tier               11148 non-null  float64
 4   CC_Contacted_LY         11158 non-null  float64
 5   Payment                 11151 non-null  object
 6   Gender                  11152 non-null  object
 7   Service_Score           11162 non-null  float64
 8   Account_user_count      11148 non-null  object
 9   account_segment         11163 non-null  object
 10  CC_Agent_Score          11144 non-null  float64
 11  Marital_Status          11048 non-null  object
 12  rev_per_month           11158 non-null  object
 13  Complain_ly             10903 non-null  float64
 14  rev_growth_yoy          11260 non-null  object
 15  coupon_used_for_payment 11260 non-null  object
 16  Day_Since_CC_connect    10903 non-null  object
 17  cashback                10789 non-null  object
 18  Login_device            11039 non-null  object
dtypes: float64(5), int64(2), object(12)
memory usage: 1.6+ MB
```

Table 2:- Dataset Information

```
Number of rows: 11260
Number of columns: 19
```

Fig 1:- Shape of dataset

Now, let us check the basic measures of descriptive statistics: This shows description of variation in various statistical measurements across variables which denotes that each variable is unique and different.

	count	unique	top	freq	mean	std	min	25%	50%	75%	max
AccountID	11260.0	NaN	NaN	NaN	25629.5	3250.62635	20000.0	22814.75	25629.5	28444.25	31259.0
Churn	11260.0	NaN	NaN	NaN	0.168384	0.374223	0.0	0.0	0.0	0.0	1.0
Tenure	11158.0	38.0	1.0	1351.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
City_Tier	11148.0	NaN	NaN	NaN	1.653929	0.915015	1.0	1.0	1.0	3.0	3.0
CC_Contacted_LY	11158.0	NaN	NaN	NaN	17.867091	8.853269	4.0	11.0	16.0	23.0	132.0
Payment	11151	5	Debit Card	4587	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Gender	11152	4	Male	6328	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Service_Score	11162.0	NaN	NaN	NaN	2.902526	0.725584	0.0	2.0	3.0	3.0	5.0
Account_user_count	11148.0	7.0	4.0	4569.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
account_segment	11163	7	Super	4062	NaN	NaN	NaN	NaN	NaN	NaN	NaN
CC_Agent_Score	11144.0	NaN	NaN	NaN	3.066493	1.379772	1.0	2.0	3.0	4.0	5.0
Marital_Status	11048	3	Married	5860	NaN	NaN	NaN	NaN	NaN	NaN	NaN
rev_per_month	11158.0	59.0	3.0	1746.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Complain_ly	10903.0	NaN	NaN	NaN	0.285334	0.451594	0.0	0.0	0.0	1.0	1.0
rev_growth_yoy	11260.0	20.0	14.0	1524.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
coupon_used_for_payment	11260.0	20.0	1.0	4373.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Day_Since_CC_connect	10903.0	24.0	3.0	1816.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
cashback	10789.0	5693.0	155.62	10.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Login_device	11039	3	Mobile	7482	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Table 3:- Describing Dataset

Insights

- AccountID is all unique values and hence it can be set as the index number
- Churn - The percentile values indicate that the tagret variable is highly imbalanced where large number of customers have not churned. This is a positive take for the management.
- Tenure - The average tenure of account is 11 months and the 75th percentile is 16 months. Tenure has a wide range of 0 to 99 months.
- City_Tier - On an average 50% of the customers are from Tier1 cities and about 25% of them from Tier3 cities
- CC_Contacted_LY - On an average 17 contacts are made by all the customers of the account in the past year. Minimum being a 4 and maximum being 132. The maximum number might be outliers.
- Service_Score - The average satisfaction score is 2.9 whereas the 50th and 75th percentile are both 3. The minimum being a 0 and maximum being a 5.
- Account_user_count - On an average 3 customers are tagged to a given account whereas there are a maximum of 6 customers tagged.
- CC_Agent_Score - The average customer service score given by a customer is 3, minimum being 1 and maximum is 5.

- rev_per_month - The average revenue generated by the account in the past year is 6.36 thousands in INR. The 75th percentile is 7 thousand whereas the maximum value is 140 thousands which indicates skewness in the data.
- Complain_ly - The average value is 0.29. All values below 50th percentile is 0 and the 75th percentile is a 1 i.e., only about 25% of the times a complaint was raised by an account in the past year.
- rev_growth_yoy - On an average the revenue growth percenatge of the account in the past year is 16.19%, minimum being 4% and maximum being 28%
- coupon_used_for_payment - Customers have used coupons on an average of ~2 times in the past year. Minimum is 0 whereas maximum is 16 times.
- Day_Since_CC_connect - The average number of days since no customers in the account has contacted the customer care is 4, the maximum is 47 days
- cashback - Average monthly cashback generated by the account in the past year is ~196 INR. 75th percentile is 200 and maximum value is 1997, which might be outliers.
- Missing values in all object type variables
- Payment - 5 payment methods, where Debit card is the most frequently used mode of payment
- Gender - Male customers are largely primary customer of the account
- account_segment - 5 different account segments on the basis of spend, with Regular Plus being the most frequent
- Marital_Status - ~53% of the customers are Married
- Login_device - 3 different types of login devices with Mobile being used most frequently

Missing Values check: The dataset contains missing values in several variables, with Except variables “AccountID”, “Churn”, “rev_growth_yoy” and “coupon_used_for_payment” all other variables have null values present.

Tenure	102
City_Tier	112
CC_Contacted_LY	102
Payment	109
Gender	108
Service_Score	98
Account_user_count	112
account_segment	97
CC_Agent_Score	116
Marital_Status	212
rev_per_month	102
Complain_ly	357
Day_Since_CC_connect	357
cashback	471
Login_device	221
dtype: int64	

Table 4: - Showing Null Values in Dataset

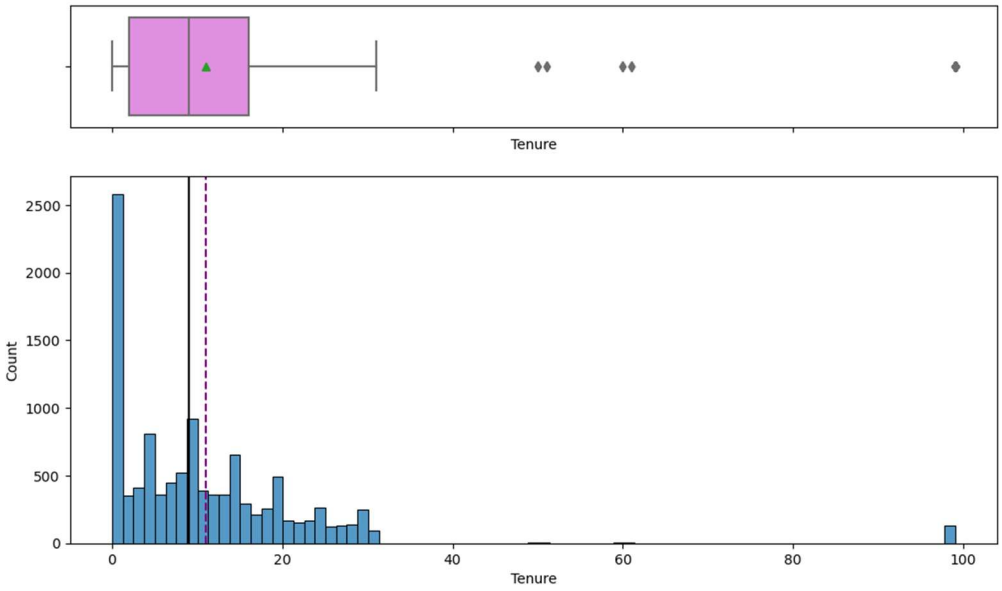
Exploratory Data Analysis:

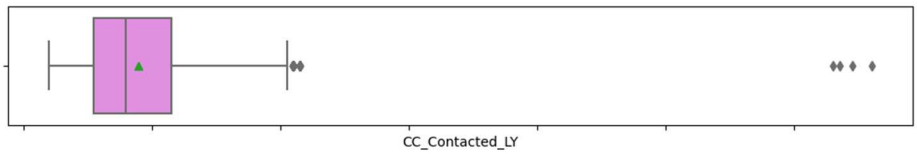
Note: We take only a few plots in this report because, as per the instruction, "**Try to keep your report within 25 pages**." All plots are in a notebook code file.

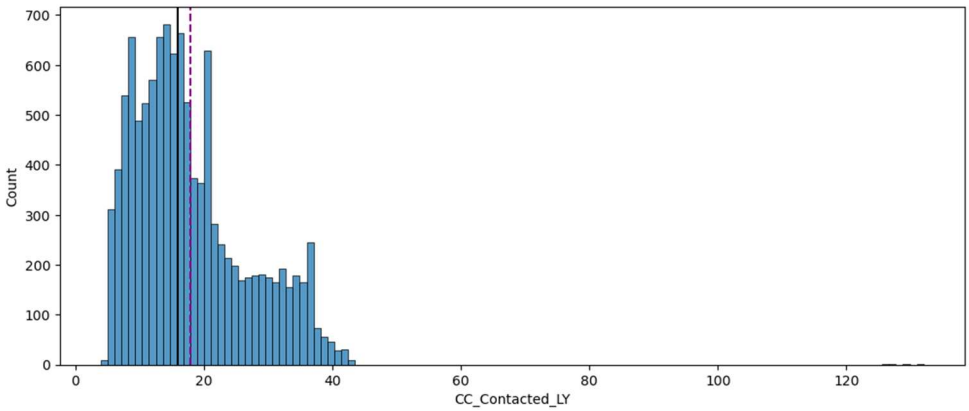
Univariate analysis (distribution and spread for every continuous attribute, distribution of data in categories for categorical ones)

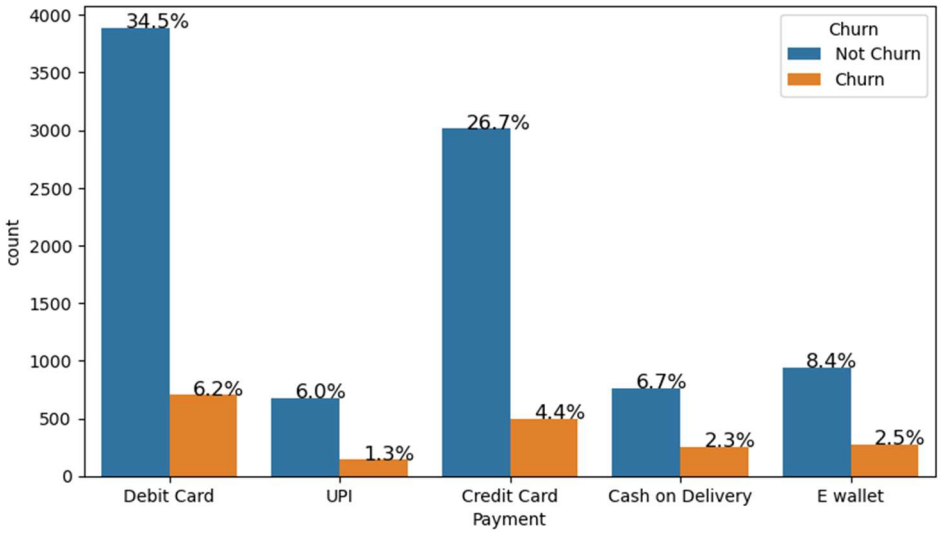
	Kurtosis	Skewness
AccountID	-1.20	0.00
Churn	1.14	1.77
Tenure	23.37	3.90
City_Tier	-1.40	0.74
CC_Contacted_LY	8.23	1.42
Service_Score	-0.67	0.00
Account_user_count	0.59	-0.39
CC_Agent_Score	-1.12	-0.14
rev_per_month	86.96	9.09
Complain_ly	-1.10	0.95
rev_growth_yoy	-0.22	0.75
coupon_used_for_payment	9.10	2.58
Day_Since_CC_connect	5.33	1.27
cashback	81.11	8.77

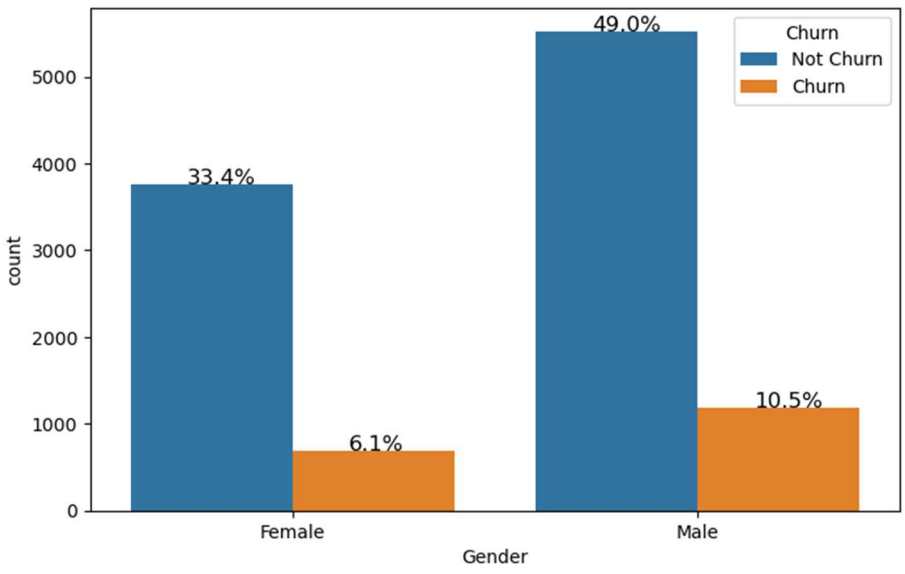
Table 5:- Showing skewness and kurtosis




CC_Contacted_LY


Count
CC_Contacted_LY


Churn
Not Churn
Churn
count
34.5%
6.2%
6.0%
1.3%
26.7%
4.4%
6.7%
2.3%
8.4%
2.5%
Debit Card
UPI
Credit Card
Cash on Delivery
E wallet
Payment


Churn
Not Churn
Churn
count
33.4%
6.1%
49.0%
10.5%
Female
Male
Gender

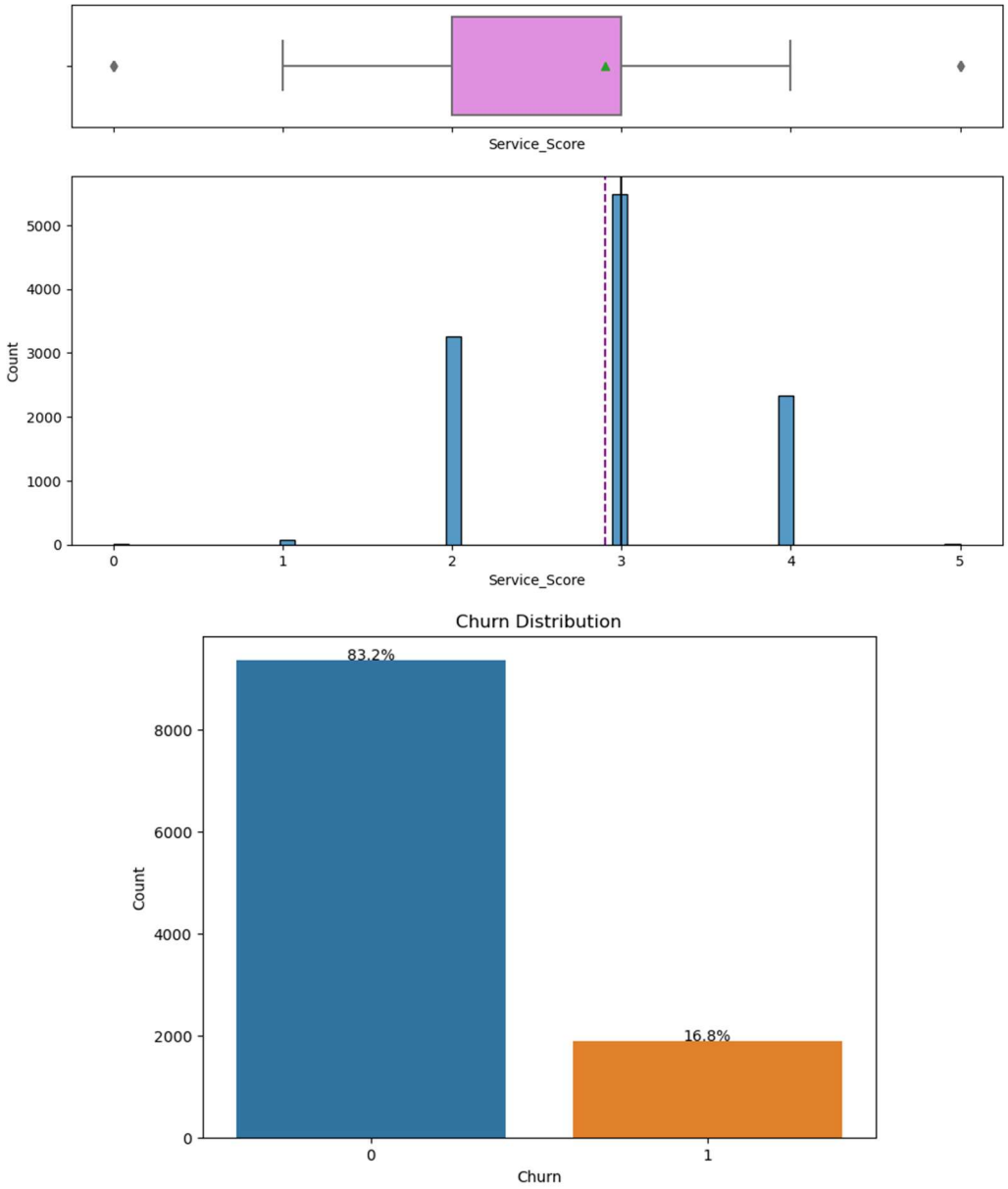


Fig 2: - Univariate analysis some plots

Inferences

- Maximum customers are form city tire type “1”, which indicates the high number of population density in this city type.
- Maximum number of customers prefer debit and credit card as their preferred mode of payment.
- The ratio of male customers are higher when compared to female.
- Average service score given by a customer for the service provided is around “3” which shows the area of improvements.
- Most of the customers are into “Super+” segment and least number of customers are into “Regular” segment.
- Most of the customers availing servives are “Married”.
- Most of the customer perfer “Mobile” as the device to avail services.

- 16.8% of customers have churned whereas 83.2% of them have not i.e., this is an imbalanced classification problem.

b) Bivariate analysis (relationship between different variables, correlations)

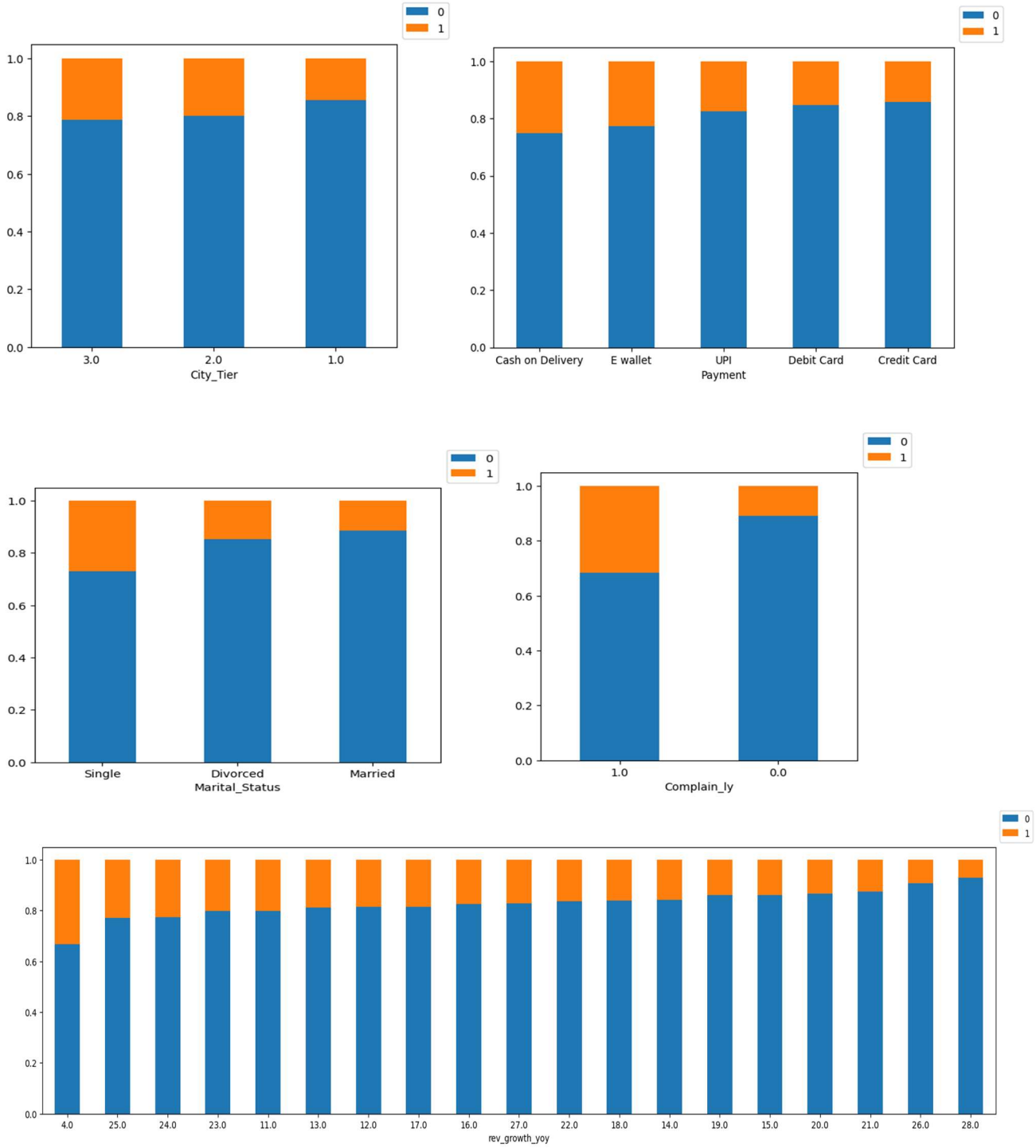


Fig 3: - Bivariate analysis some plots

Inferences

- City_tier "1" has shown maximum churning when compared with "2" and "3".

- Customer with prefreffred mode of payment as “debit card” and “credit card” are more prone to churn.
- Customers with gender as “Male” are showing more Churn ratio as compared to female.
- Customers into “Regular Plus” segment showing more churn.
- Single customers are more tend to churn when compared with divorced and married.
- Customers using the service over mobile shows more churn.

c) **Multi-variate analysis to understand relationship b/w variables**

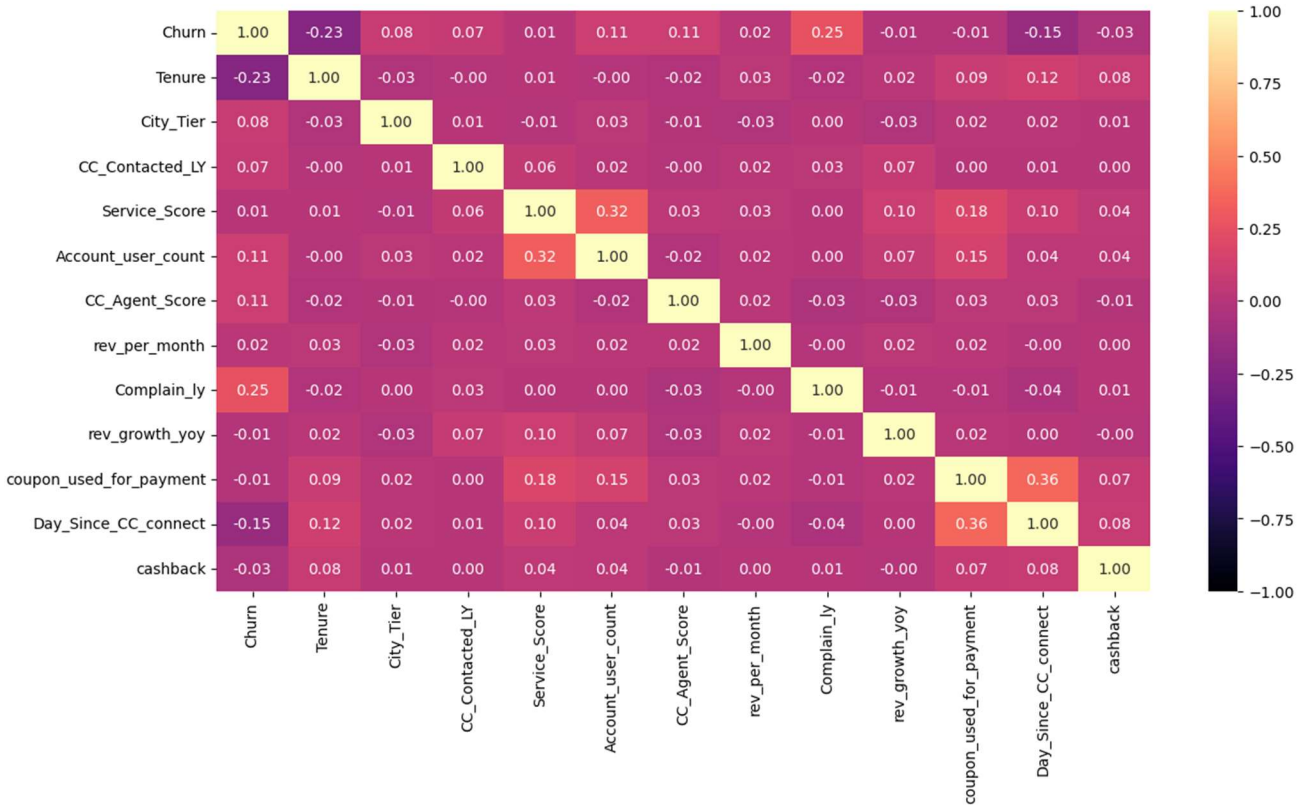


Fig 4: - Multivariate analysis some plots

Inferences

- Churn is in moderate positive correlation with Complain_ly, Account_user_count, CC_Agent_Score and negative correlation with Tenure, Day_Since_CC_connect
- Service_Score is in moderate positive correlation with Account_user_count, coupon_used_for_payment and in weak positive correlation with rev_growth_yoy, Day_Since_CC_connect
- Account_user_count is in moderate positive correlation with Service_Score, coupon_used_for_payment and Churn
- rev_growth_yoy is in weak positive correlation with Service_Score
- coupon_used_for_payment is in moderate positive correlation with Day_Since_CC_connect, Service_Score and Account_user_count
- City_Tier, CC_Contacted_LY, rev_per_month, rev_growth_yoy and cashback are in a weak

+ve/-ve correlation with all the other variables

- Overall, there is no strong positive/negative correlation among the variables.
- Lower the tenure with the company, higher the chances that a customer will churn
- Customers from city tier 1 and 3 seems to churn more comparatively
- Customers will less number of contacts with the customer care in the past year have churned more in number
- Potential are the customers who have given a service satisfaction scores 3,2 and 4
- Accounts tagged with 4,3 and 5 customers are customers who have churned
- The customers who have given an agent score of 3 and above show the churn pattern
- Account that generate lower average monthly revenue and average revenue growth percentage (last year vs the previous yaer) are potential to churn
- There is no distinct churn pattern with reference to the complaints received past year
- Lower the number of times coupons were used for payment, higher the chances that the customer will churn
- Customers with lower average monthly cash back and those contacting the customer care frequently have higher chances to churn

Key Insights from EDA

Customer churn is more common among:

Accounts:

- Those with a tenure of less than approximately 5 months, indicating new customers.
- Also, regular Plus account holders are more likely to churn.
- Additionally, accounts with lower revenue growth percentage in the past year compared to the previous year.
- Those that generated an average monthly cashback under 200 INR, are more prone to churning.

Customers:

- Customers from City Tier1 followed by City Tier3 are more likely to churn.
- Additionally, customers who prefer using Debit Card for payments and Mobile for login, those who frequently contact customer care (an average of ~19 times in the past year), and those who are Male and Single are at higher risk of churn.
- Moreover, customers who give a service satisfaction score of 3 or less and an agent score of 3 or more, who raised a complaint in the past year, and who used coupons for payment less than 2 times are more likely to churn.

Data Cleaning and Pre-processing

Removal of unwanted variables (if applicable): After in-depth understanding of data we conclude that removal of variables is not required at this stage of project. We can remove the variable “AccountID” which denotes a unique ID assigned to unique customers. However, removing them will lead to 8 duplicate rows. Rest all the variables looks important looking at the univariate and bi-variate analysis.

Variable transformation: Before normalizing or scaling the variables, it's important to note that some machine learning algorithms, like decision trees and random forests, are invariant to the scale of the features. However, for algorithms like logistic regression and k-nearest neighbors, feature scaling is crucial for optimal performance.

- We see that the different variable has different dimensions. Like variable “Cashback” denotes currency where as “CC_Agent_Score” denotes rating provided by the customers. Due to which they differ in their statistical rating as well.
- Scaling would be required for this data set which in turn will normalize the date and standard deviation will be close to “0”.
- Using MinMax scalar to perform normalization of data.

Churn	0.374223	Churn	0.374223
Tenure	12.879782	Tenure	0.130099
City_Tier	0.915015	City_Tier	0.457508
CC_Contacted_LY	8.853269	CC_Contacted_LY	0.069166
Payment	1.379118	Payment	0.344780
Gender	0.489684	Gender	0.489684
Service_Score	0.725584	Service_Score	0.145117
Account_user_count	1.022976	Account_user_count	0.204595
account_segment	1.272367	account_segment	0.318092
CC_Agent_Score	1.379772	CC_Agent_Score	0.344943
Marital_Status	0.896791	Marital_Status	0.448396
rev_per_month	11.909686	rev_per_month	0.085681
Complain_ly	0.451594	Complain_ly	0.451594
rev_growth_yoy	3.757721	rev_growth_yoy	0.156572
coupon_used_for_payment	1.969551	coupon_used_for_payment	0.123097
Day_Since_CC_connect	3.697637	Day_Since_CC_connect	0.078673
cashback	178.660514	cashback	0.089464
Login_device	0.575374	Login_device	0.287687
dtype: float64		dtype: float64	

Fig 5: Before and after Normalization

Outlier treatment:

Churn	16.838366
Tenure	1.234458
City_Tier	0.000000
CC_Contacted_LY	0.373002
Service_Score	0.115453
Account_user_count	6.758437
CC_Agent_Score	0.000000
rev_per_month	1.642984
Complain_ly	0.000000
rev_growth_yoy	0.000000
coupon_used_for_payment	12.255773
Day_Since_CC_connect	0.293073
cashback	7.806394
dtype: float64	

Table 6: percentage of outliers in each column

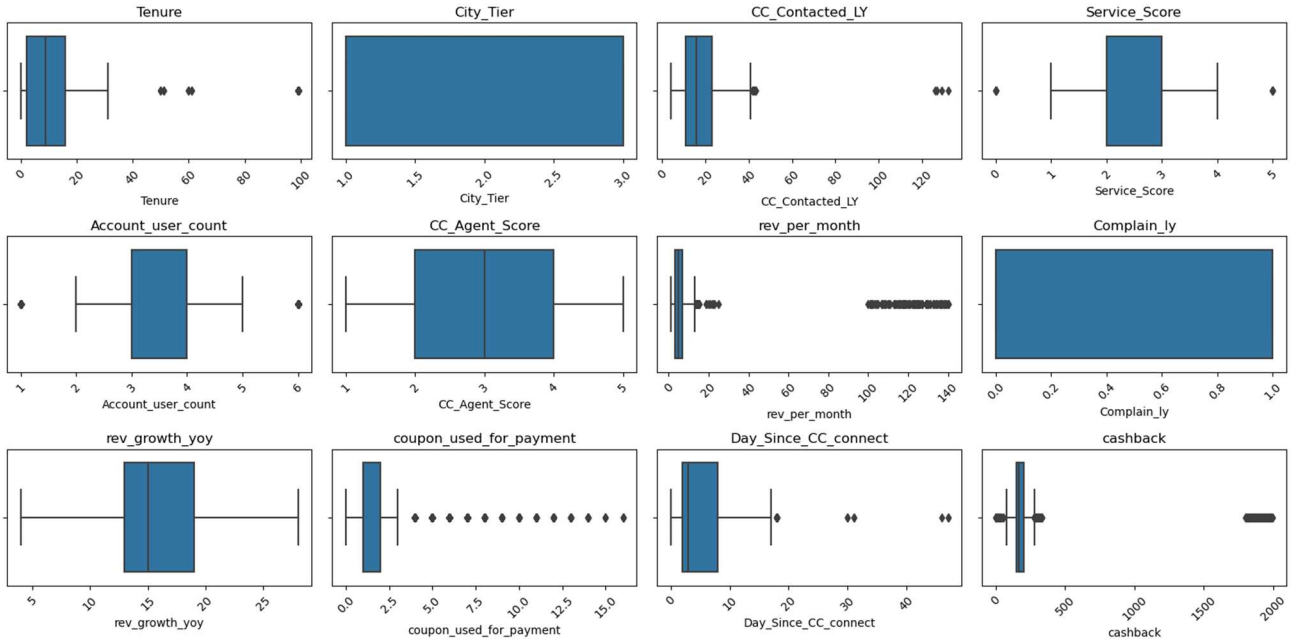


Fig 6: Boxplot of outliers

Total Number of Outliers: 5328 and Percentage of Outliers: 3.639841508402787. As seen above, most of the numerical columns have outliers. However, I choose not to treat them, as in real case scenario the data will have outliers and would want the model to learn the underlying pattern for such customers.

Missing Value treatment: Out of 19 variables we have data anomalies present in 17 variable and null values in 15 variables. Apply KNN Imputer to treat the missing values.

Churn	0
Tenure	218
City_Tier	112
CC_Contacted_LY	102
Payment	109
Gender	108
Service_Score	98
Account_user_count	444
account_segment	97
CC_Agent_Score	116
Marital_Status	212
rev_per_month	791
Complain_ly	357
rev_growth_yoy	3
coupon_used_for_payment	3
Day_Since_CC_connect	358
cashback	473
Login_device	221
dtype: int64	

Table 7: Missing Values in each columns

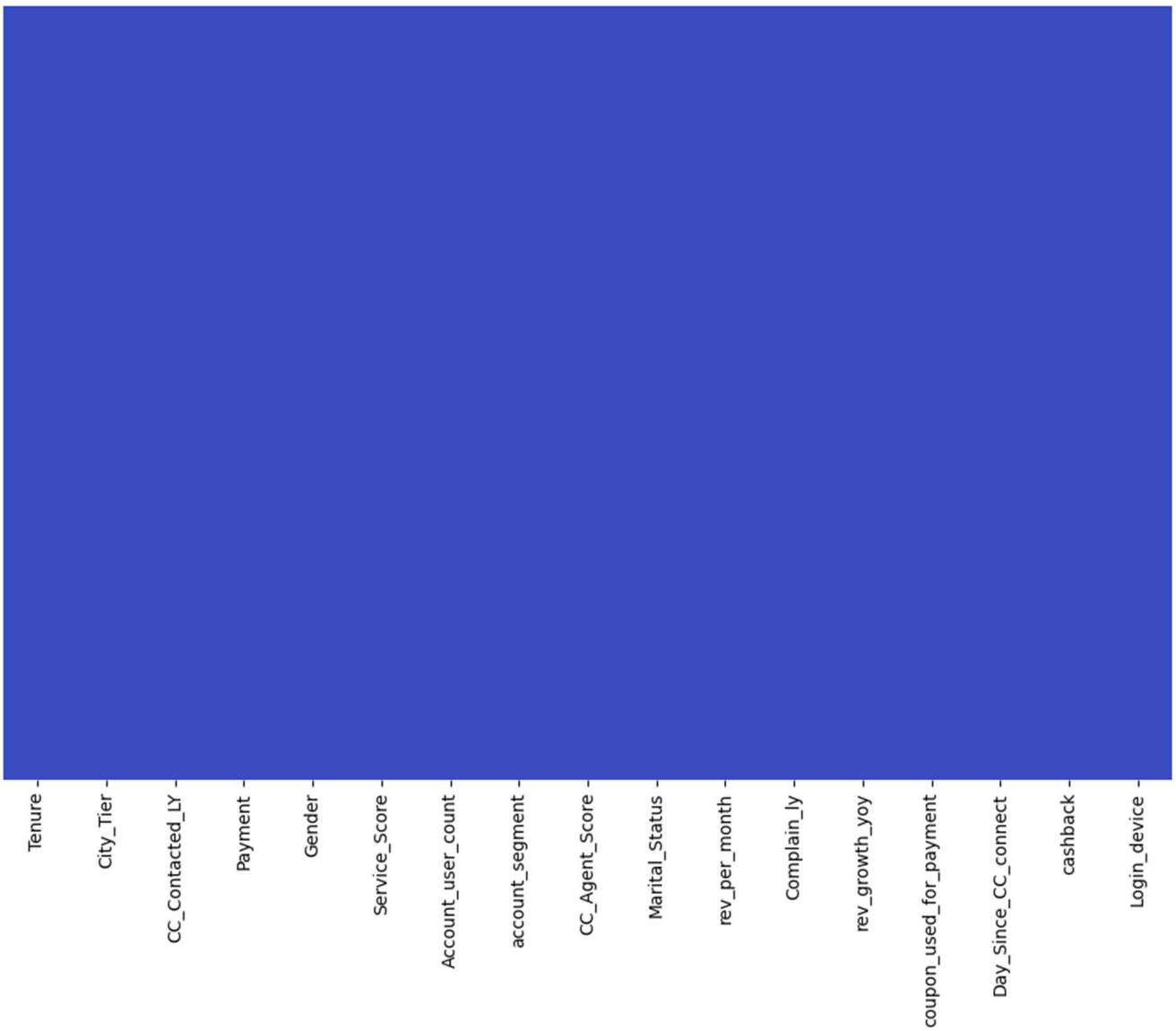


Fig 7: Heat map of After Missing Values Treatment

Feature Extraction

- From the EDA it was clearly seen that the churn rate for Marital_Status, Payment, account_segment and Login_device w.r.t Male customers was higher than that of Female customers. Hence creating interaction variables for the same.
- Also, the churn pattern for Account_user_count varied w.r.t Service_Score. Hence creating a ratio variable for the same.
- Train and test sets now have 29 columns each and 7882 rows in Train set and 3378 rows in Test set.
- We inverse the label encoding for Payment, Gender, account_segment, Marital_Status and Login_device.

Feature Engineering:

- We use the Label encoding the features Payment, account_segment and Login_device.

- Creating dummy varibales for other categorical features

	Tenure	City_Tier	CC_Contacted_LY	Payment	Service_Score	Account_user_count	account_segm
8274	13.0	1.0	14.0	2	2.0	3.0	
5259	19.0	1.0	8.0	0	4.0	4.0	
7756	0.0	1.0	10.0	1	2.0	4.0	
2399	22.0	1.0	13.0	2	2.0	3.0	
5820	16.0	1.0	18.0	2	2.0	3.0	

5 rows × 30 columns

	Tenure	City_Tier	CC_Contacted_LY	Payment	Service_Score	Account_user_count	account_segm
6926	20.0	1.0	12.0	1	3.0	3.0	
1669	3.0	1.0	34.0	1	3.0	3.0	
9498	1.0	3.0	12.0	1	3.0	4.0	
3287	16.0	1.0	7.0	2	3.0	5.0	
2973	29.0	1.0	27.0	0	3.0	4.0	

5 rows × 30 columns

Table 8: Train and Test dataset

Model Building

From the above visual and non-visual analysis, we can conclude that it's a case of classification model, where in the target variable needs to be classified into "Yes" or "No". As a data analyst we have below algorithms to build the desired mechanism to predict if a given customer will churn or not: -

Splitting Data into Train and Test Dataset: - We have divided data into Train and Test dataset into 70:30 ratio and building various models on training dataset and testing for accuracy over testing dataset.

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

Fig 8: Shape of Training and Test Dataset

Let's build various models including Logistic Regression, LDA, KNN, Decision Tree Classifier, Naive Bayes, Bagging, Boosting (AdaBoost & Gradient Boosting), Random Forest Classifier, and XGBClassifier. We'll use hyperparameter tuning where applicable, evaluate models using AUC score and ROC curve, and employ SMOTE (Synthetic Minority Over-sampling Technique) for dealing with class imbalance if necessary.

Hyper parameter tuning

- The model seems to perform reasonably well in terms of accuracy on both the training and testing datasets, achieving an accuracy of around 84%.
- However, there are significant differences in precision, recall, and F1-score between the two classes (0 and 1), indicating potential class imbalance issues.
- The recall for class 1 (positive class) is particularly low, suggesting that the model has difficulty correctly identifying instances of class 1, leading to a low F1-score for this class.
- By fine-tuning the hyper parameters, we can improve the model's ability to generalize and make accurate predictions on new, unseen data.

1. Logistic Regression: We start with Logistic Regression because it's simple yet effective for binary classification problems like churn prediction. It gives us a good baseline to compare other models.

2. LDA (Linear Discriminant Analysis): LDA is chosen because it's similar to Logistic Regression but assumes the predictors are normally distributed. If our data meets this assumption, LDA can perform well and give insights into how the predictors are related to the target.

3. KNN (K-Nearest Neighbors): KNN is chosen because it's intuitive and doesn't make strong assumptions about the data. It works by finding the 'k' nearest data points to a new observation and making predictions based on their class. This flexibility can be useful if the relationship between predictors and target is not linear.

4. Decision Tree Classifier: Decision trees are chosen because they can capture complex relationships in the data and are easy to interpret. They partition the data into segments based on predictor values and make predictions accordingly.

5. Naive Bayes: Naive Bayes is chosen because it's simple, fast, and works well with high-dimensional data. It assumes independence between predictors, which might not be true in reality, but often performs surprisingly well in practice.

6. Bagging: Bagging (Bootstrap Aggregating) is chosen because it improves the stability and accuracy of machine learning algorithms. It works by creating multiple bootstrap samples of the training data and building a model on each sample. Then, it combines the predictions of all models to make a final prediction.

7. Boosting (AdaBoost & Gradient Boosting): Boosting algorithms are chosen because they sequentially build weak learners and combine them to create a strong learner. AdaBoost focuses on minimizing errors by giving more weight to misclassified observations, while Gradient Boosting builds new models that complement the errors of previous models.

8. Random Forest Classifier: Random Forest is chosen because it's an ensemble method that combines multiple decision trees to reduce overfitting and improve predictive performance. It works well with large

datasets and high-dimensional data.

9. XGBClassifier (XGBoost): XGBoost is chosen because it's an optimized implementation of Gradient Boosting. It's highly efficient, scalable, and often outperforms other algorithms in predictive accuracy.

In summary, each model is chosen based on its strengths and suitability for the churn prediction task. We aim to explore a variety of algorithms to see which one(s) perform best for our specific dataset and problem.

```
Cross-Validation Performance :

LR : 38.941826796525405
LDA : 30.29744669649908
KNN : 42.768564827001256
Dtree : 84.00777982510019
Naive Bayes : 40.62823550057033
Bagging : 79.33871486648533
RandomForest : 80.71744026205727
Adaboost : 57.532976514287384
GBM : 60.288379983036464
XGBoost : 86.8389342224562

Test set Performance :

LR : 43.2937181663837
LDA : 34.63497453310696
KNN : 47.70797962648557
Dtree : 84.88964346349745
Naive Bayes : 42.95415959252971
Bagging : 83.5314091680815
RandomForest : 87.26655348047538
Adaboost : 60.611205432937176
GBM : 61.62988115449915
XGBoost : 88.11544991511036
```

Table 9: Cross validated score of all model

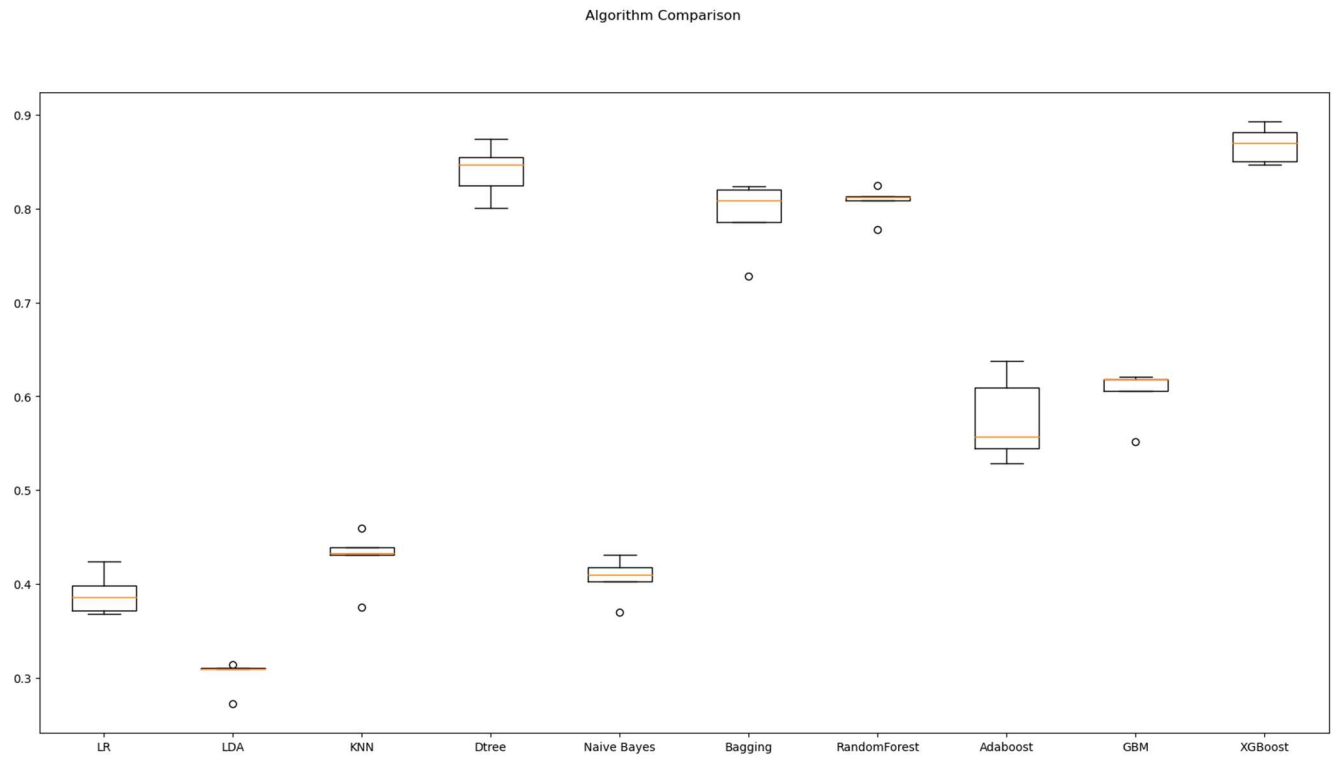


Fig 9: Comparison of all models

Insights:

- XGBoost is giving the highest mean cross-validated Recall followed by RandomForest, DecisionTree and Bagging classifiers
- Highest validation set Recall achieved is 88 by XGBoost.
- Will build the above models, hypertune and SMOTE them to get much better results

We build various models individual like Logistic Regression, LDA, KNN, Decision Tree Classifier, Naive Bayes, Bagging, Boosting (AdaBoost & Gradient Boosting), Random Forest Classifier, and XGBClassifier. We'll use hyperparameter tuning where applicable, evaluate models using AUC score and ROC curve, and employ SMOTE (Synthetic Minority Over-sampling Technique) for dealing with class imbalance if necessary.

After all models performance, we get our best-fit model. Performance scores (like accuracy, precision, recall, F1 score, and AUC score for train and test datasets) are all in the below table.

Model Evaluation Criterion

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.

	Training Dataset - 70%							
Model Name	Accuracy	Precision-0	Recall-0	F1 Score-0	Precision-1	Recall-1	F1 Score-1	AUC Score
Logistic Regression	87.07	0.88	0.98	0.93	0.77	0.32	0.45	0.854
Logistic Regression GridSearchC	87.65	0.89	0.98	0.93	0.76	0.38	0.5	0.86
Logistic Regression SMOTE	78.45	0.8	0.76	0.78	0.77	0.8	0.79	0.858
Linear Discriminant Analysis	86.29	0.87	0.98	0.92	0.73	0.28	0.4	0.853
LDA GridSearchCV	86.37	0.87	0.98	0.92	0.73	0.28	0.41	0.822
LDA SMOTE	62.85	0.58	0.98	0.72	0.93	0.28	0.43	0.824
KNN Model	94.35	0.95	0.98	0.97	0.89	0.76	0.82	0.981
KNN GridSearchCV	1	1	1	1	1	1	1	1
KNN SMOTE	87.48	0.97	0.78	0.86	0.81	0.97	0.89	0.974
Decision Tree Classifier	1	1	1	1	1	1	1	1
Decision Tree GridSearchCV	84.21	0.94	0.87	0.9	0.52	0.72	0.6	0.794
Decision Tree Classifier SMOTE	1	1	1	1	1	1	1	1
Naive Bayes	80.96	0.88	0.89	0.89	0.42	0.38	0.4	0.758
Naive Bayes GridSearchCV	70.04	0.73	0.64	0.68	0.68	0.77	0.72	0.773
Random Forest	1	1	1	1	1	1	1	1
Random Forest SMOTE	1	1	1	1	1	1	1	1
Bagging	99.47	0.99	1	1	1	0.97	0.98	1
Bagging SMOTE	99.85	1	1	1	1	1	1	1
Ada-Boost	89.81	0.92	0.96	0.94	0.76	0.57	0.65	0.921
Ada-Boost SMOTE	87.92	0.87	0.89	0.88	0.88	0.87	0.88	0.953
Gradient Boosting	91.19	0.93	0.98	0.95	0.84	0.63	0.72	0.952
Gradient Boosting SMOTE	92.54	0.92	0.93	0.93	0.93	0.92	0.93	0.979
XGBoost Classifier	99.94	1	1	1	1	1	1	1
XGBoost Classifier GridSearchC	92.69	0.9	1	0.95	1	0.88	0.94	0.979
XGBoost Classifier SMOTE	92.54	0.92	0.93	0.93	0.93	0.92	0.93	0.979

Fig 10: Compares ion Table of All models for Train Data set

	Testing Dataset - 30%							
Model Name	Accuracy	Precision-0	Recall-0	F1 Score-0	Precision-1	Recall-1	F1 Score-1	AUC Score
Logistic Regression	87.51	0.88	0.98	0.93	0.83	0.36	0.5	0.854
Logistic Regression GridSearch(	87.83	0.89	0.98	0.93	0.79	0.41	0.54	0.859
Logistic Regression SMOTE	76.49	0.95	0.76	0.84	0.41	0.8	0.54	0.853
Linear Discriminant Analysis	87	0.87	0.98	0.93	0.81	0.33	0.47	0.853
LDA GridSearchCV	86.94	0.87	0.98	0.93	0.81	0.33	0.47	0.818
LDA SMOTE	87	0.87	0.98	0.93	0.81	0.33	0.47	0.817
KNN Model	89.84	0.92	0.96	0.94	0.77	0.6	0.67	0.918
KNN GridSearchCV	95.7	0.98	0.97	0.97	0.86	0.9	0.88	0.934
KNN SMOTE	76.37	0.97	0.74	0.84	0.42	0.89	0.57	0.905
Decision Tree Classifier	92.8	0.96	0.95	0.96	0.79	0.8	0.8	0.879
Decision Tree GridSearchCV	83.89	0.93	0.87	0.9	0.53	0.71	0.61	0.789
Decision Tree Classifier SMOTE	91.17	0.96	0.93	0.95	0.72	0.81	0.76	0.874
Naive Bayes	81.82	0.88	0.9	0.89	0.48	0.44	0.46	0.757
Naive Bayes GridSearchCV	64.94	0.92	0.63	0.75	0.3	0.76	0.43	0.765
Random Forest	96.44	0.97	0.99	0.98	0.96	0.84	0.89	0.99
Random Forest SMOTE	96.62	0.98	0.98	0.98	0.92	0.88	0.9	0.987
Bagging	95.38	0.95	0.99	0.97	0.95	0.78	0.85	0.984
Bagging SMOTE	95.61	0.97	0.97	0.97	0.87	0.87	0.87	0.982
Ada-Boost	89.57	0.92	0.96	0.94	0.76	0.59	0.66	0.907
Ada-Boost SMOTE	85.73	0.94	0.88	0.91	0.57	0.73	0.64	0.891
Gradient Boosting	90.85	0.92	0.97	0.95	0.81	0.62	0.7	0.932
Gradient Boosting SMOTE	88.48	0.94	0.92	0.93	0.65	0.72	0.69	0.913
XGBoost Classifier	**96.65**	**0.97**	**0.99**	**0.98**	**0.93**	**0.87**	**0.9**	**0.991**
XGBoost Classifier GridSearchC	91.88	0.98	0.92	0.95	0.71	0.9	0.79	0.97
XGBoost Classifier SMOTE	88.48	0.94	0.92	0.93	0.65	0.72	0.69	0.913

Fig 11: Compares ion Table of All models for Test Data set

To determine the best-fit model among the options provided, we need to consider several factors including accuracy, precision, recall, F1-score, and AUC score. Let's analyze the models:

From the above tabular representation of all the scores for training and testing dataset across various model we can conclude that the XGBoost Classifier model with default values of hyper-parameters is best optimized for the given dataset. (Highlighted in BOL and Yellow colour)

Best-fit Model:

- The XGBoost Classifier stands out as the best-fit model. It achieved near-perfect accuracy on both the training and testing datasets (99.94% and 96.65% respectively), along with high precision, recall, F1-score, and AUC score.
- XGBoost has demonstrated robust performance across multiple evaluation metrics, indicating its

effectiveness in predicting churn.

Overfitting:

- Overfitting occurs when a model performs exceptionally well on the training data but poorly on unseen data.
- The KNN Model with GridSearchCV achieved 100% accuracy on the training dataset, suggesting potential overfitting. However, its performance on the testing dataset was also high (95.7%), indicating good generalization.
- Decision Tree Classifier and Random Forest also achieved 100% accuracy on the training dataset, which may signal overfitting. However, their performance on the testing dataset was also high, suggesting effective generalization.
- Bagging and Bagging SMOTE models achieved extremely high accuracy (99.47% and 99.85% respectively) on the training dataset, indicating potential overfitting. However, their performance on the testing dataset was also very high, suggesting good generalization.

Underfitting:

- Underfitting occurs when a model is too simple to capture the underlying patterns in the data.
- Linear Discriminant Analysis (LDA) models, especially those trained with SMOTE, exhibited lower accuracy, precision, recall, and F1-score compared to other models. This suggests that these models may not be capturing the complexities of the data adequately, indicating potential underfitting.
- Naive Bayes models, particularly Naive Bayes GridSearchCV, also demonstrated lower performance metrics compared to other models, indicating potential underfitting.

Conclusion:

- Overall, the XGBoost Classifier emerges as the best-fit model due to its high accuracy, precision, recall, F1-score, and AUC score on both the training and testing datasets.
- While some models may exhibit signs of overfitting or underfitting, their performance on the testing dataset suggests reasonable generalization, making them suitable choices depending on the specific requirements and constraints of the problem.

Feature importance of XGBoost (Tuned)

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

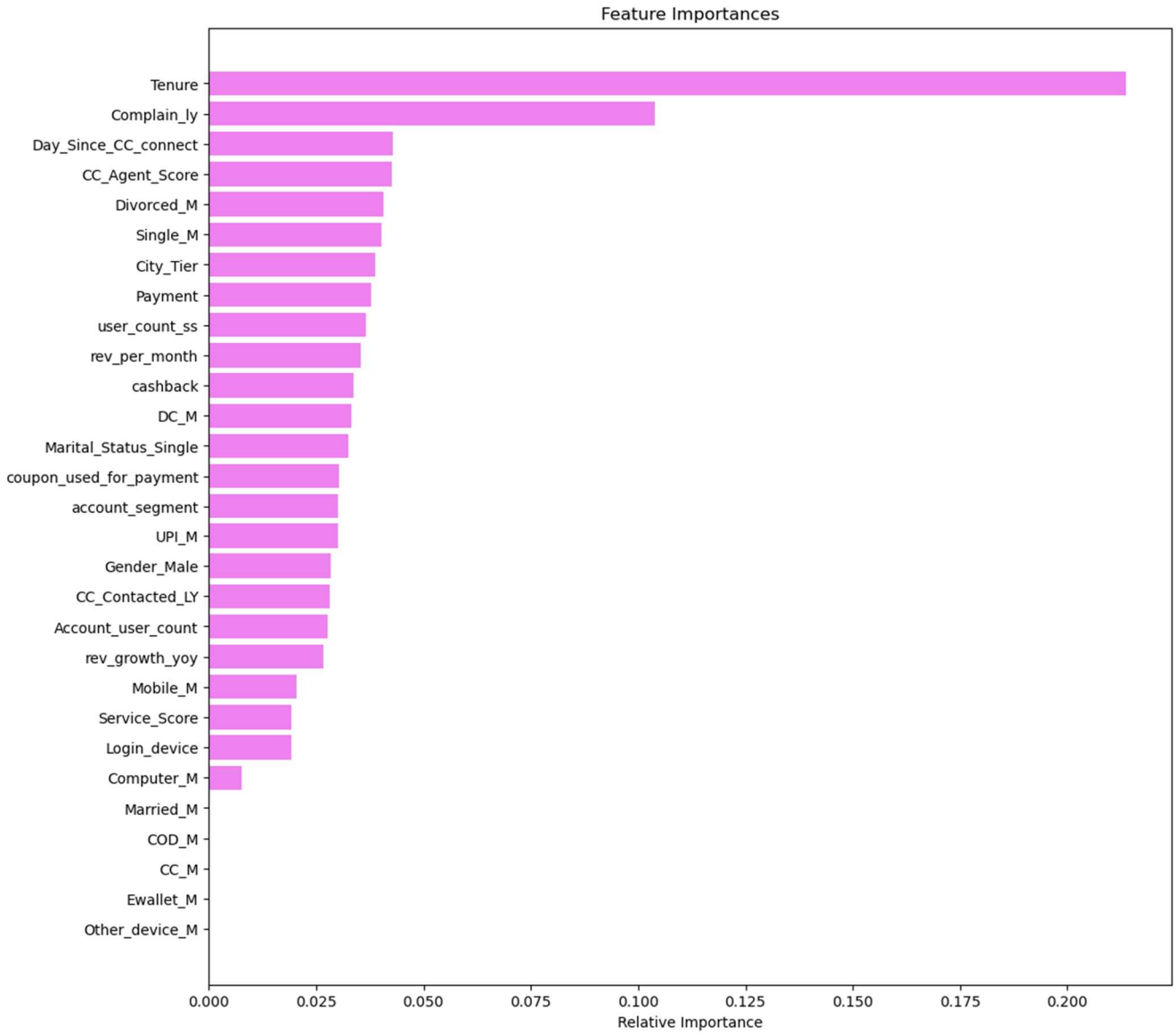


Fig 12: Feature importance of XGBoost (Tuned)

Model Validation

When it comes to model validation of a classification problem statement we cannot just get relied on accuracy, we need to look at various others parameter like F1 score, Recall, precision, ROC curve and AUC score, along with confusion matrix. The details of these parameters are described below:

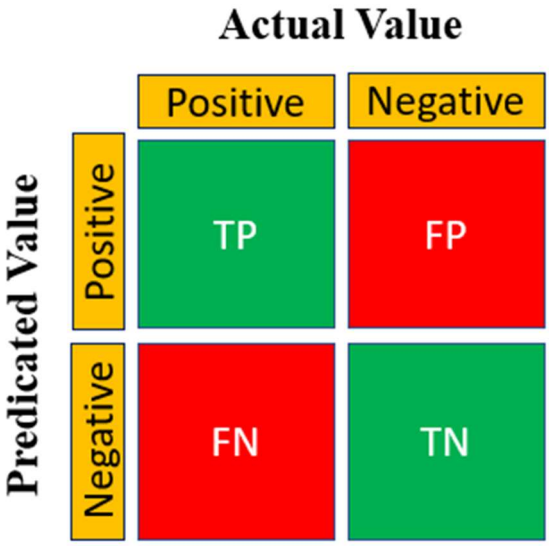


Fig 13: Confusion matrix

Confusion Matrix: Confusion Matrix usually causes a lot of confusion even in those who are using them regularly. Terms used in defining a confusion matrix are TP, TN, FP, and FN.

True Positive (TP): The actual is positive in real and at the same time the prediction was classified correctly.

False Positive (FP): The actual was actually negative but was falsely classified as positive. True Negative: - The actuals were actually negative and was also classified as negative which is the right thing to do.

False Negative: The actuals were actually positive but was falsely classified as negative.

Various Components of Classification Report:

Accuracy: This term tells us how many right classifications were made out of all the classifications.

Accuracy = (TP + TN) / (TP + FP +TN + FN)

Precision: Out of all that were marked as positive, how many are actually truly positive.

Precision = TP / (TP + FP)

Recall or Sensitivity: Out of all the actual real positive cases, how many were identified as positive.

Recall = TP/ (TN + FN)

Specificity: Out of all the real negative cases, how many were identified as negative.

Specificity = TN/ (TN + FP)

F1-Score: As we saw above, sometimes we need to give weightage to FP and sometimes to FN. F1 score is a weighted average of Precision and Recall, which means there is equal importance given to FP and FN. This is a very useful metric compared to "Accuracy". The problem with using accuracy is that if we have a highly imbalanced dataset for training (for example, a training dataset with 95% positive class

and 5% negative class), the model will end up learning how to predict the positive class properly and will not learn how to identify the negative class. But the model will still have very high accuracy in the test dataset too as it will know how to identify the positives really well.

F1 score = 2* (Precision * Recall) / (Precision + Recall)

Area Under Curve (AUC) and ROC Curve: AUC or Area Under Curve is used in conjecture with ROC Curve which is Receiver Operating Characteristics Curve. AUC is the area under the ROC Curve. So, let's first understand the ROC Curve.

A ROC Curve is drawn by plotting TPR or True Positive Rate or Recall or Sensitivity (which we saw above) in the y-axis against FPR or False Positive Rate in the x-axis. FPR = 1- Specificity (which we saw above).

TPR = TP/ (TP + FN)

FPR = 1 – TN/ (TN+FP) = FP/ (TN + FP)

When we want to select the best model, we want a model that is closest to the perfect model. In other words, a model with AUC close to 1. When we say a model has a high AUC score, it means the model's ability to separate the classes is very high (high separability). This is a very important metric that should be checked while selecting a classification model.

Final interpretation / Recommendation

Insights from Analysis

1. Tenure Insights:

- Customers with shorter tenure, especially less than a month, are more likely to churn.
- Management should focus on retaining new customers by providing incentives or personalized onboarding experiences.

2. City Tier Analysis:

- Churn rates vary across city tiers, with tier 1 and tier 3 cities showing higher churn rates.
- Targeted retention strategies should be implemented for customers in these cities to improve retention.
- Customer Care Contact Frequency:
- Customers who contact customer care frequently are more likely to churn.
- Proactive customer support and personalized follow-ups can help in addressing customer concerns and reducing churn.

3. Service Satisfaction Scores:

- Customers giving service satisfaction scores of 3, 2, and 4 are potential churners.
- Efforts should be made to improve service quality and address customer feedback to enhance satisfaction and retention.

4. Account Tagging:

- Accounts tagged with 4, 3, and 5 customers are associated with churn.
- Monitoring and addressing issues specific to these accounts can help in retaining customers.

5. Revenue Generation and Growth:

- Accounts with lower average monthly revenue and revenue growth percentage are likely to churn.
- Offering tailored promotions or loyalty programs may incentivize customers to increase spending and reduce churn.

6. Coupon Usage and Cashback:

Customers using coupons infrequently and receiving lower average monthly cashback are prone to churn. Promotional offers and rewards targeted at increasing coupon usage and cashback can enhance customer engagement and retention.

7. Login Device Preference:

Customers predominantly using computers for login show higher churn rates.
Improving the user experience on other devices or offering exclusive benefits for mobile or smart TV users may help in retention.

8. Complaint Handling:

While no distinct churn pattern is observed based on complaints received, prompt resolution and follow-up on complaints can prevent customer attrition.

Overall Insights:

- Implement proactive retention strategies for new customers with shorter tenure.
- Tailor services and incentives based on city tier preferences.
- Enhance customer care support and address feedback promptly to improve satisfaction.
- Monitor and address issues specific to tagged accounts to prevent churn.
- Encourage higher spending and engagement through targeted promotions and rewards.
- Improve user experience on preferred login devices to retain customers.
- Focus on complaint resolution and follow-up to maintain customer satisfaction and loyalty.
- By implementing these recommendations, the management can effectively reduce churn rates and improve overall customer retention, leading to long-term business growth and success.

Insights from Model Building

1. The XGBoost classifier emerges as the top performer among all evaluated models.
2. It demonstrates exceptional accuracy, precision, recall, and AUC scores on both the training and testing datasets.
3. The model shows strong predictive capability, accurately identifying instances of both churn and non-churn with high accuracy.
4. There are no signs of overfitting or underfitting observed in the XGBoost classifier, indicating consistent and high performance across different evaluation metrics and cross-validation scores.
5. Hyperparameter tuning further enhances the model's performance, improving accuracy and

generalization capability.

6. Feature importance analysis highlights key factors influencing customer churn, including tenure, complaints in the last year, and days since CC connection.
7. Understanding these influential factors can help businesses implement targeted interventions to reduce churn and retain customers effectively.
8. Overall, the XGBoost classifier offers a powerful and reliable solution for predicting customer churn, with strong performance across various evaluation metrics.
9. Its ability to generalize well to unseen data and provide insights into important features makes it a valuable tool for businesses aiming to mitigate customer churn and enhance retention strategies.

Business Insights and Recommendations

- The business can use this model to identify customers who may churn
- Top five features that drive the attrition are Tenure, Complain_ly, Marital_Status_Single, account_segment and Single_M
- Business may provide introductory offers to attract new customers and exclusive offers to existing new customers
- 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.
- 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
- Provide targeted offers to Female customers who prefer E-wallet/Mobile, from the Regular account segments
- Also provide exclusive family offers for Married customers from HNI segment as churn rate is higher among them
- Business may consider increasing the cashback to Regular Plus customers and Debit card payment in order to reduce the churn
- Exit interview can be conducted to get feedback from outgoing customers and work towards the betterment of the services provided.
- Businesses can encourage current customers to refer their friends or family to gain new customers. They can team up with other lifestyle businesses to give out vouchers to both new and loyal customers.
- Businesses can divide their customers based on how much they spend, like those who always look for deals or those who want the best prices, and use different ways to get more customers for each group.
- They can give loyal customers free cloud storage. They can also send personalized emails to important customers to make them feel special.
- To give the best service, they can have a special team just for the most important customers, so they don't have to wait for help.
- Businesses can learn about their customers and send them small gifts on special days. Writing a thank-you note by hand on the bill can also make customers happy.
- They should check back with customers who had problems and ask for feedback. They can also do surveys to see if customers are happy.

- It's important for businesses to fix any problems customers have quickly. They can also encourage people to use their own e-wallet by giving discounts.
- Businesses should offer special deals for single people because they might leave more often. They can also make a plan that includes everything a family needs.
- They should try to get more customers in smaller cities. Finally, they can promote paying through bank accounts or UPI, which is easy and safe.

Appendix

Codes Snapshot: -

```
from sklearn.model_selection import StratifiedKFold, cross_val_score
from sklearn.metrics import recall_score
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from xgboost import XGBClassifier

from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

```
# Replacing the invalid data with np.nan

Customer_Churn["Tenure"].replace("#", np.nan, inplace=True)
Customer_Churn["Account_user_count"].replace("@", np.nan, inplace=True)
Customer_Churn["rev_per_month"].replace("+", np.nan, inplace=True)
Customer_Churn["rev_growth_yoy"].replace("$", np.nan, inplace=True)
Customer_Churn["coupon_used_for_payment"].replace(["#", "$", "*"], np.nan, inplace=True)
Customer_Churn["Day_Since_CC_connect"].replace("$", np.nan, inplace=True)
Customer_Churn["cashback"].replace("$", np.nan, inplace=True)
```

```
# Defining a method to plot histogram and boxplot combined in a single plot
def histogram_boxplot(Customer_Churn, feature, figsize=(12, 7), kde=False, bins=None):
    """
    Boxplot and histogram combined
    feature : dataframe column
    figsize : size of figure (default (12,7))
    kde : whether to show the density curve (default False)
    bins : number of bins (default None / auto)

    """
    f2, (ax_box2, ax_hist2) = plt.subplots(
        nrows=2,  # Number of rows of the subplot grid = 2
        sharex=True,  # x-axis will be shared among all subplots
        gridspec_kw={"height_ratios": (0.25, 0.75)},
        figsize=figsize,
    )  # creating the 2 subplots

    sns.boxplot(data=Customer_Churn, x=feature, ax=ax_box2, showmeans=True, color="violet")

    sns.histplot(
        data=Customer_Churn, x=feature, kde=kde, ax=ax_hist2, bins=bins
    ) if bins else sns.histplot(
        data=Customer_Churn, x=feature, kde=kde, ax=ax_hist2
    )  # For histogram

    ax_hist2.axvline(Customer_Churn[feature].mean(), color="purple", linestyle="--")  # Add mean to the histogram

    ax_hist2.axvline(Customer_Churn[feature].median(), color="black", linestyle="-")  # Add median to the histogram
```

```python
# Defining a method to plot histogram and boxplot combined in a single plot
def histogram_boxplot(Customer_Churn, feature, figsize=(12, 7), kde=False, bins=None):
    """
    Boxplot and histogram combined
    feature : dataframe column
    figsize : size of figure (default (12,7))
    kde : whether to show the density curve (default False)
    bins : number of bins (default None / auto)

    """
    f2, (ax_box2, ax_hist2) = plt.subplots(
        nrows=2,  # Number of rows of the subplot grid = 2
        sharex=True,  # x-axis will be shared among all subplots
        gridspec_kw={"height_ratios": (0.25, 0.75)},
        figsize=figsize,
    )  # creating the 2 subplots

    sns.boxplot(data=Customer_Churn, x=feature, ax=ax_box2, showmeans=True, color="violet")

    sns.histplot(
        data=Customer_Churn, x=feature, kde=kde, ax=ax_hist2, bins=bins
    ) if bins else sns.histplot(
        data=Customer_Churn, x=feature, kde=kde, ax=ax_hist2
    )  # For histogram

    ax_hist2.axvline(Customer_Churn[feature].mean(), color="purple", linestyle="--")  # Add mean to the histogram

    ax_hist2.axvline(Customer_Churn[feature].median(), color="black", linestyle="-")  # Add median to the histogram
```

```python
#To perform scaling and normalization on the dataset using Min-Max scaling
from sklearn.preprocessing import MinMaxScaler
Customer_Churn['Scaled_Churn'] = MinMaxScaler().fit_transform(Customer_Churn[['Churn']])
Customer_Churn['Scaled_Tenure'] = MinMaxScaler().fit_transform(Customer_Churn[['Tenure']])
Customer_Churn['Scaled_City_Tier'] = MinMaxScaler().fit_transform(Customer_Churn[['City_Tier']])
Customer_Churn['Scaled_CC_Contacted_LY'] = MinMaxScaler().fit_transform(Customer_Churn[['CC_Contacted_LY']])
Customer_Churn['Scaled_Payment'] = MinMaxScaler().fit_transform(Customer_Churn[['Payment']])
Customer_Churn['Scaled_Gender'] = MinMaxScaler().fit_transform(Customer_Churn[['Gender']])
Customer_Churn['Scaled_Service_Score'] = MinMaxScaler().fit_transform(Customer_Churn[['Service_Score']])
Customer_Churn['Scaled_Account_user_count'] = MinMaxScaler().fit_transform(Customer_Churn[['Account_user_count']])
Customer_Churn['Scaled_account_segment'] = MinMaxScaler().fit_transform(Customer_Churn[['account_segment']])
Customer_Churn['Scaled_CC_Agent_Score'] = MinMaxScaler().fit_transform(Customer_Churn[['CC_Agent_Score']])
Customer_Churn['Scaled_Marital_Status'] = MinMaxScaler().fit_transform(Customer_Churn[['Marital_Status']])
Customer_Churn['Scaled_rev_per_month'] = MinMaxScaler().fit_transform(Customer_Churn[['rev_per_month']])
Customer_Churn['Scaled_Complain_ly'] = MinMaxScaler().fit_transform(Customer_Churn[['Complain_ly']])
Customer_Churn['Scaled_rev_growth_yoy'] = MinMaxScaler().fit_transform(Customer_Churn[['rev_growth_yoy']])
Customer_Churn['Scaled_coupon_used_for_payment'] = MinMaxScaler().fit_transform(Customer_Churn[['coupon_used_for_payment']])
Customer_Churn['Scaled_Day_Since_CC_connect'] = MinMaxScaler().fit_transform(Customer_Churn[['Day_Since_CC_connect']])
Customer_Churn['Scaled_cashback'] = MinMaxScaler().fit_transform(Customer_Churn[['cashback']])
Customer_Churn['Scaled_Login_device'] = MinMaxScaler().fit_transform(Customer_Churn[['Login_device']])
```

```
def Married_M(df):
    if df["Marital_Status"] == "Married" and df["Gender"] == "Male":
        return 1
    else:
        return 0


X_train["Married_M"] = X_train.apply(lambda X_train: Married_M(X_train), axis=1)

X_test["Married_M"] = X_test.apply(lambda X_test: Married_M(X_test), axis=1)


def Single_M(df):
    if df["Marital_Status"] == "Single" and df["Gender"] == "Male":
        return 1
    else:
        return 0


X_train["Single_M"] = X_train.apply(lambda X_train: Single_M(X_train), axis=1)

X_test["Single_M"] = X_test.apply(lambda X_test: Single_M(X_test), axis=1)


def Divorced_M(df):
    if df["Marital_Status"] == "Divorced" and df["Gender"] == "Male":
        return 1
    else:
        return 0


X_train["Divorced_M"] = X_train.apply(lambda X_train: Divorced_M(X_train), axis=1)

X_test["Divorced_M"] = X_test.apply(lambda X_test: Divorced_M(X_test), axis=1)
```

```
# Label encoding the features Payment, account_segment and Login_device

le = LabelEncoder()

X_train["Payment"] = le.fit_transform(X_train["Payment"])
X_test["Payment"] = le.transform(X_test["Payment"])

X_train["account_segment"] = le.fit_transform(X_train["account_segment"])
X_test["account_segment"] = le.transform(X_test["account_segment"])

X_train["Login_device"] = le.fit_transform(X_train["Login_device"])
X_test["Login_device"] = le.transform(X_test["Login_device"])
```

```
models = []  # Empty list to store all the models

# Appending the models to the list
models.append(("LR", LogisticRegression(solver="newton-cg", random_state=1)))
models.append(("LDA", LinearDiscriminantAnalysis()))
models.append(("KNN", KNeighborsClassifier()))
models.append(("Dtree", DecisionTreeClassifier(random_state=1)))
models.append(("Naive Bayes", GaussianNB()))
models.append(("Bagging", BaggingClassifier(random_state=1)))
models.append(("RandomForest", RandomForestClassifier(random_state=1)))
models.append(("Adaboost", AdaBoostClassifier(random_state=1)))
models.append(("GBM", GradientBoostingClassifier(random_state=1)))
models.append(("XGBoost", XGBClassifier(random_state=1, eval_metric="logloss")))


results = []  # Empty list to store all models' CV scores
names = []  # Empty list to store name of the models

# Looping through all the models to get the mean cross validated score
print("Cross-Validation Performance :\n")
for name, model in models:
    scoring = "recall"
    kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)  # Setting the number of splits to 5
    cv_result = cross_val_score(estimator=model, X=X_train, y=y_train, scoring=scoring, cv=kfold)
    results.append(cv_result)
    names.append(name)
    print("{} : {}".format(name, cv_result.mean() * 100))

print("\nTest set Performance :\n")
for name, model in models:
    model.fit(X_train, y_train)
    scores = recall_score(y_test, model.predict(X_test)) * 100
    print("{} : {}".format(name, scores))
```

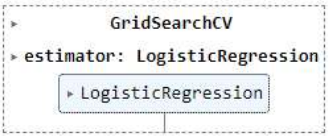
Hyperparameter tuning

- The model seems to perform reasonably well in terms of accuracy on both the training and testing datasets, achieving an accuracy of around 84%.
- However, there are significant differences in precision, recall, and F1-score between the two classes (0 and 1), indicating potential class imbalance issues.
- The recall for class 1 (positive class) is particularly low, suggesting that the model has difficulty correctly identifying instances of class 1, leading to a low F1-score for this class.
- By fine-tuning the hyperparameters, we can improve the model's ability to generalize and make accurate predictions on new, unseen data.

```
# loading GridSearchCV and creating dataframe for parameters
from sklearn.model_selection import GridSearchCV
param_grid = {
    'solver': ['lbfgs','newton-cg', 'liblinear', 'sag', 'saga'],
    'penalty': ['l1','l2','none'],
    'tol':[0.0001,0.00001]
}

grid_search = GridSearchCV(estimator = log_reg, param_grid = param_grid, cv = 10, n_jobs=-1,scoring='f1')
```

```
# fitting grid search into traing dataset
grid_search.fit(X_train, y_train)
```



Apply SMOTE

```
from collections import Counter
from sklearn.datasets import make_classification
from imblearn.over_sampling import SMOTE
```

```
smote = SMOTE(random_state=1)
X_train_res, y_train_res = smote.fit_resample(X_train, y_train)
```

```
#fitting model into training dataset
log_reg.fit(X_train_res, y_train_res)
```

```
LogisticRegression
LogisticRegression(random_state=1)
```

Finding the right value for n_neighbor

```
# getting the ideal number of value of "N"
# empty list that will hold accuracy scores
ac_scores = []

# perform accuracy metrics for values from 1,3,5....19
for k in range(1,20,2):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    # evaluate test accuracy
    scores = knn.score(X_test, y_test)
    ac_scores.append(scores)

# changing to misclassification error
MCE = [1 - x for x in ac_scores]
MCE
```

```
importances = rf.feature_importances_
indices = np.argsort(importances)
feature_names = list(X_train.columns)

plt.figure(figsize=(12, 12))
plt.title("Feature Importances")
plt.barh(range(len(indices)), importances[indices], color="violet", align="center")
plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
plt.xlabel("Relative Importance")
plt.show()
```

----------END----------