CAPSTONE REPORT - CUSTOMER CHURN – FINAL REPORT

By Vinish Vincent

Great Learnings
PGP-DSBA - 23

Table of content

Sr. No	Content	Page No
1	Introduction of the business problem	2-3
2	EDA and Business Implication	3-11
3	Data Cleaning and Pre-processing	11-14
4	Model building	15-25
5	Model validation	25-26
6	Final interpretation / recommendation	26-27

1. Introduction of the business problem

1.1 Problem Statement

The task at hand involves creating a churn prediction model for a DTH provider facing intense market competition. The primary goal is to identify potential account churns, considering that a single account may encompass multiple customers. Losing an account translates to losing multiple customers, making account retention crucial.

As the assigned individual responsible for developing the churn prediction model, the objective is to provide insightful business recommendations for targeted campaigns. These campaigns should be unique and offer clear, valuable incentives to potential churners. It is essential to avoid recommending overly generous or subsidized offers that might lead to financial losses for the company. The recommendations should undergo scrutiny by the revenue assurance team, and if they perceive a risk of significant financial loss, the approval may be withheld. Therefore, the campaign suggestions must strike a balance between customer retention and financial prudence.

1.2 Need of the study/project

The need for the churn prediction project arises from the highly competitive environment that a DTH (Direct-to-Home) provider is currently facing. In such a competitive landscape, retaining existing customers becomes a significant challenge. The project aims to address this challenge by developing a predictive model capable of identifying potential churners among the company's accounts.

The goal is not only to predict churn but also to provide segmented offers to potential churners. This requires a tailored approach based on customer behaviour, preferences, and other relevant factors. The project also aims to provide clear and effective campaign recommendations that strike a balance between customer retention and financial viability. These recommendations need to be carefully crafted to gain approval from the revenue assurance team.

1.3 Understanding business/social opportunity

The churn prediction project represents a significant business opportunity centred on enhancing customer retention and bolstering revenue growth. By harnessing the power of data-driven decision-making, the project aims to proactively identify potential churners, enabling the implementation of precisely targeted campaigns and offers. This strategic approach not only improves the company's competitive positioning in a fierce market but also contributes to long-term sustainability.

Some key opportunities include:

- **Customer Retention Boost**: Enhance strategies by identifying potential churners, preserving accounts, and ensuring cost-effective retention of existing customers.
- **Revenue Uplift**: Execute the churn prediction model to implement targeted campaigns, encouraging customer loyalty and positively impacting the company's financial performance.
- **Strategic Data Utilization**: Leverage customer data for informed decision-making, extending beyond churn prediction to enhance strategic planning and resource allocation.

- **Competitive Edge**: Actively address customer churn to gain a competitive advantage, positioning the company as responsive, engaging, and fostering a positive brand image.
- **Campaign Effectiveness**: Craft compelling campaign recommendations, striking a balance between customer attractiveness and financial viability for the company to drive engagement and loyalty.
- **Social Impact**: Beyond business benefits, improved customer retention contributes to job stability, economic sustainability, and enhanced overall customer experiences, potentially leading to positive social outcomes.

2. EDA and Business Implication

Understanding data

- Data set has 11,260 number of observations and 19 variables (18 independent and 1 dependent or target variable)
- It appears the data has been collected at random, a total of 11,260 entries with unique account ID, across gender and marital status.
- Looking at variables like "CC_Contacted_L12m", "rev_per_month", "Complain_l12m", "rev_growth_yoy", "coupon_used_l12m", "Day_Since_CC_connect" and "cashback_l12m" it seems that the data has been collected for 12 month (1 year).
- The data seems to be collected through customer interactions, transactions, and possibly surveys. The features capture a range of information, including customer demographics, transaction details, service interactions, and complaints.
- Data is mixed of categorical as well as continuous variables.

Inspection of data (rows, columns, descriptive details)

Dataset head:

	AccountID	Churn	Tenure	City_Tier	CC_Contacted_LY	Payment	Gender	Service_Score	Account_user_count	account_segment	CC_Agent_Score	Marital_Status	rev_per_month	Complain_ly	rev_growth_yoy	coupon_used_for_payment	Day_Since_CC_connect	cashback	Login_device
0	20000	1	4	3.0	6.0	Debit Card	Female	3.0	3	Super	2.0	Single	9	1.0	11	1	5	159.93	Mobile
1	20001	1	0	1.0	8.0	UPI	Male	3.0	4	Regular Plus	3.0	Single	7	1.0	15	0	0	120.9	Mobile
2	20002	1	0	1.0	30.0	Debit Card	Male	2.0	4	Regular Plus	3.0	Single	6	1.0	14	0	3	NaN	Mobile
3	20003	1	0	3.0	15.0	Debit Card	Male	2.0	4	Super	5.0	Single	8	0.0	23	0	3	134.07	Mobile
4	20004	1	0	1.0	12.0	Credit Card	Male	2.0	3	Regular Plus	5.0	Single	3	0.0	11	1	3	129.6	Mobile

Fig 1: Raw dataset

Dataset shape:

```
The number of Rows is = 11260
The number of Columns is = 19
```

Fig 2: Shape of dataset

Dataset information and data types:

```
<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
```

Fig 3: Information of dataset

- There are 2 integer columns, 5 float columns and 12 object type columns
- There appears to be missing values in all the variables except for 'Account ID', 'Churn', 'rev_growth_yoy', and 'coupon_used_for_payment'

Data Summary:

	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

Fig 4: Summary of dataset

- The 'Churn' column indicates a churn rate of approximately 16.8%, with a majority (83.2%) of customers not churning. This binary variable suggests a relatively balanced dataset.

- The 'Payment' column shows that 'Debit Card' is the most common payment method, accounting for 41% of the data. Other payment methods are present as well, but 'Debit Card' stands out as the top choice among customers.

Checking Null values:

```
AccountID                   0
Churn                       0
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
rev_growth_yoy              0
coupon_used_for_payment     0
Day_Since_CC_connect      357
cashback                  471
Login_device              221
dtype: int64
```

```
Number of duplicate rows: 0
```

Fig 5: Null value of dataset

- Except variables "AccountID", "Churn", "rev_growth_yoy" and "coupon_used_for_payment" all other variables have null values present
- There appears to be no duplicate values

Kurtosis and Skewness of data:

	Kurtosis	Skewness
AccountID	-1.200	0.000
Churn	1.142	1.773
City_Tier	-1.398	0.737
CC_Contacted_LY	8.226	1.423
Service_Score	-0.668	0.004
CC_Agent_Score	-1.125	-0.142
Complain_ly	-1.096	0.951

Fig 6: Kurtosis and Skewness of dataset

- The 'Churn' column exhibits positive skewness (1.773), indicating that the distribution is skewed to the right. This suggests that there are more customers who did not churn, aligning with the earlier observation of a higher percentage of non-churned customers.
- 'CC_Contacted_LY' demonstrates high positive kurtosis (8.226), signifying heavy tails and a more peaked distribution. This suggests a concentration of customer care interactions around the mean with some extreme values.
- 'Complain_ly' has negative skewness (0.951) and negative kurtosis (-1.096), suggesting a distribution that is skewed to the left and has lighter tails. This implies that the majority of customers had few or no complaints, with a few customers having a relatively higher number of complaints.

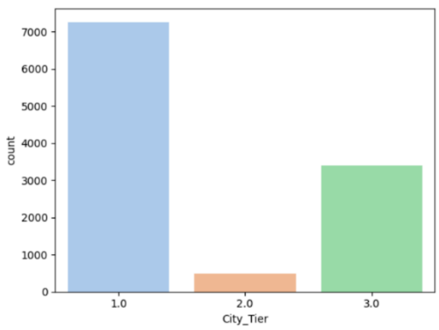
Attributes (variable info)

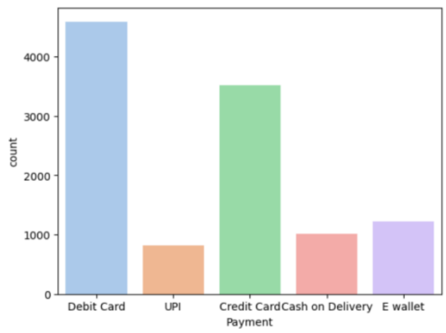
Variable information:

- **AccountID**: Unique identifier for each customer account.
- **Churn**: A binary variable (1 or 0) indicating whether the customer has churned (1) or not (0).
- **Tenure**: The duration of the customer's association with the service, possibly in months.
- **City_Tier**: The tier of the city in which the customer is located (e.g., Tier 1, Tier 2, etc.).
- **CC_Contacted_LY**: Contact made with customer care in the last year (LY).
- **Payment**: The method of payment used by the customer (Debit Card, UPI, Credit Card, etc.).
- **Gender**: Gender of the account holder.
- **Service_Score**: A score related to the service provided to the customer.
- **Account_user_count**: The number of users associated with the account.
- **Account_segment**: The segmentation of the account (e.g., Super, Regular Plus).
- **CC_Agent_Score**: Score related to customer care agent interaction.
- **Marital_Status**: Marital status of the account holder.
- **rev_per_month**: Revenue generated per month.
- **Complain_ly**: Whether the customer had a complaint in the last year (1 for Yes, 0 for No).
- **rev_growth_yoy**: Year-over-year revenue growth.
- **coupon_used_for_payment**: Number of coupons used for payment.
- **Day_Since_CC_connect**: Number of days since connecting with customer care.
- **cashback**: Cashback received by the customer.
- **Login_device**: The device used for account login (e.g., Mobile).

Exploratory data analysis

Univariate Analysis:





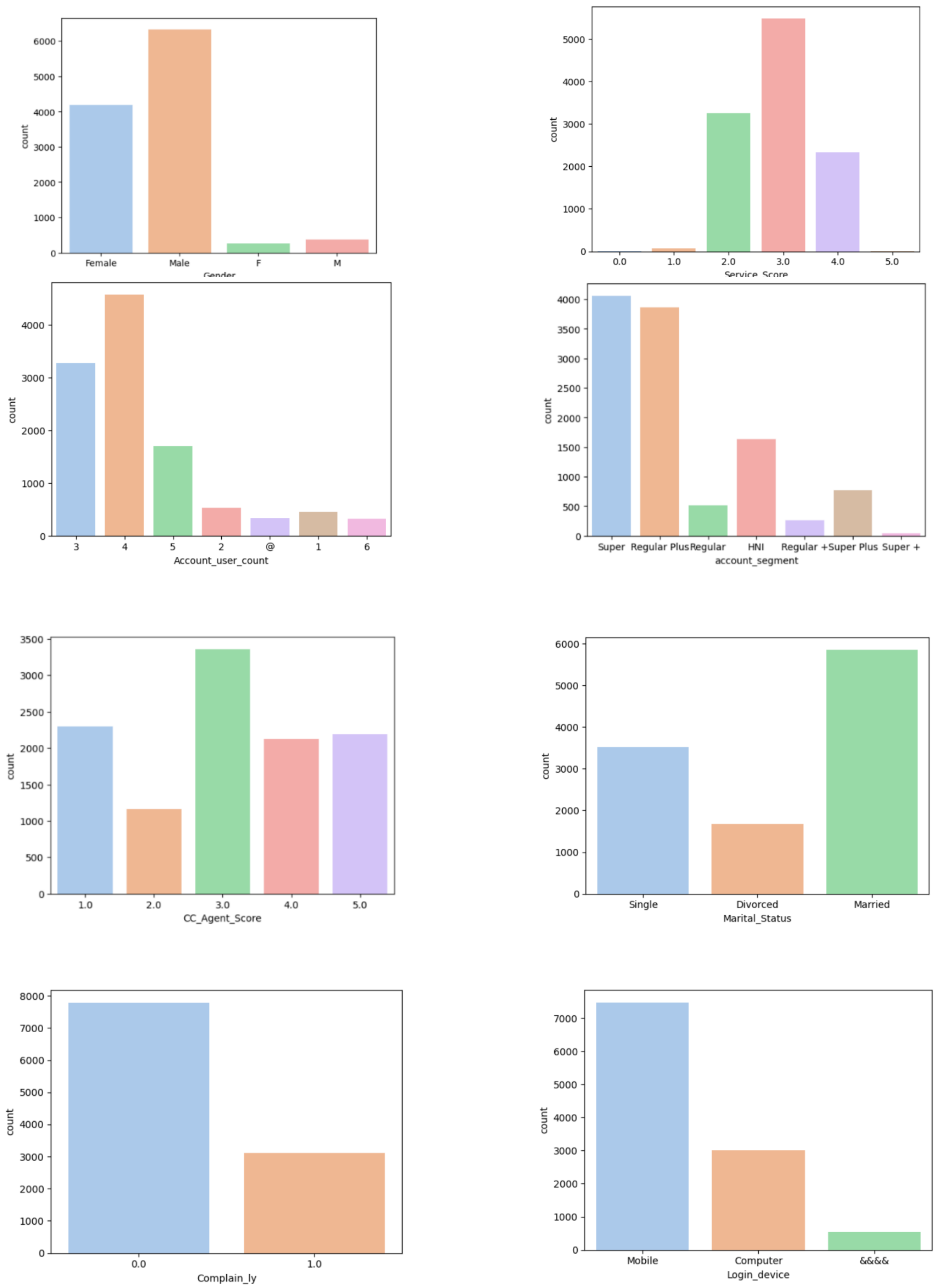


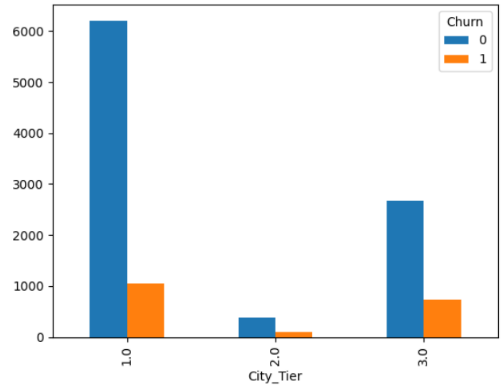
Fig 7: Univariate Analysis of dataset

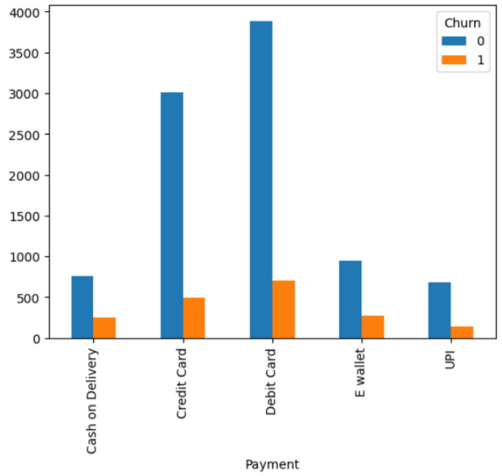
- The majority of customers prefer using debit and credit cards as their primary mode of payment
- A significant number of customers are located in City Tier 1, indicating a high population density in this city tier

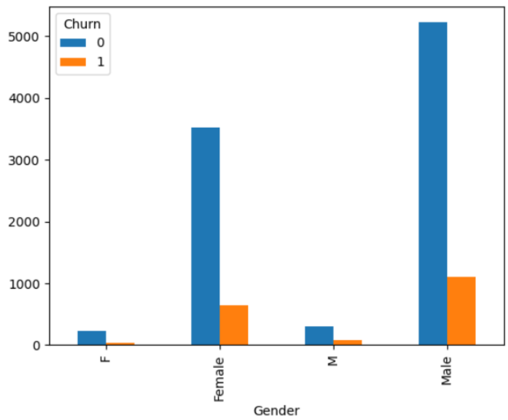
- The male-to-female ratio among customers is skewed towards males
- On average, customers provide a service score of around "3," suggesting areas for potential improvement in service quality
- The majority of customers in the dataset are married
- The preferred device for accessing services among customers is predominantly mobile

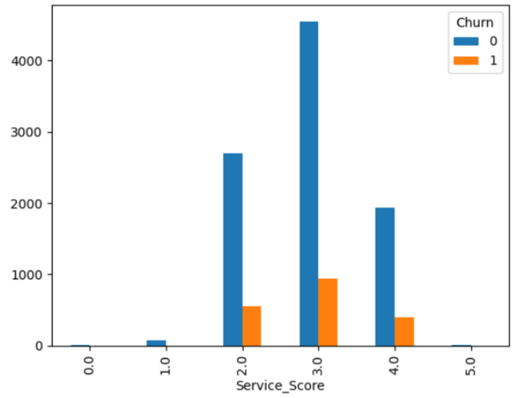
Bivariate analysis (relationship between different variables , correlations)

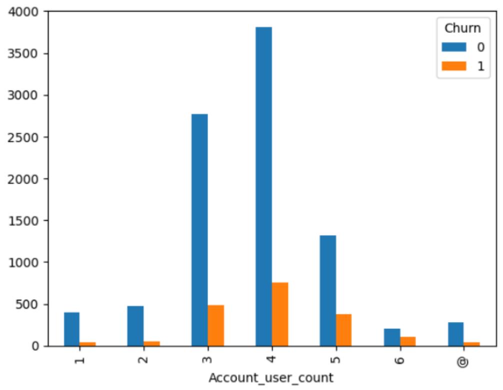
Bivariate Analysis:

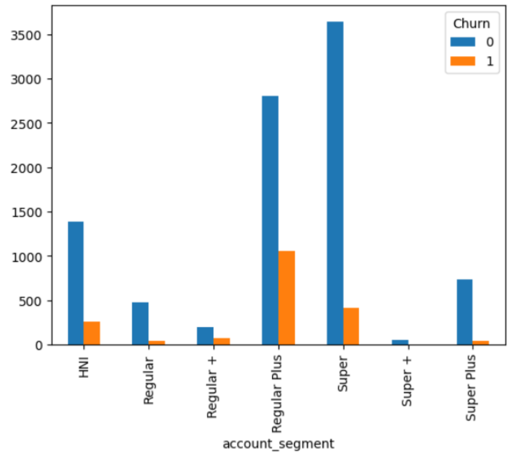












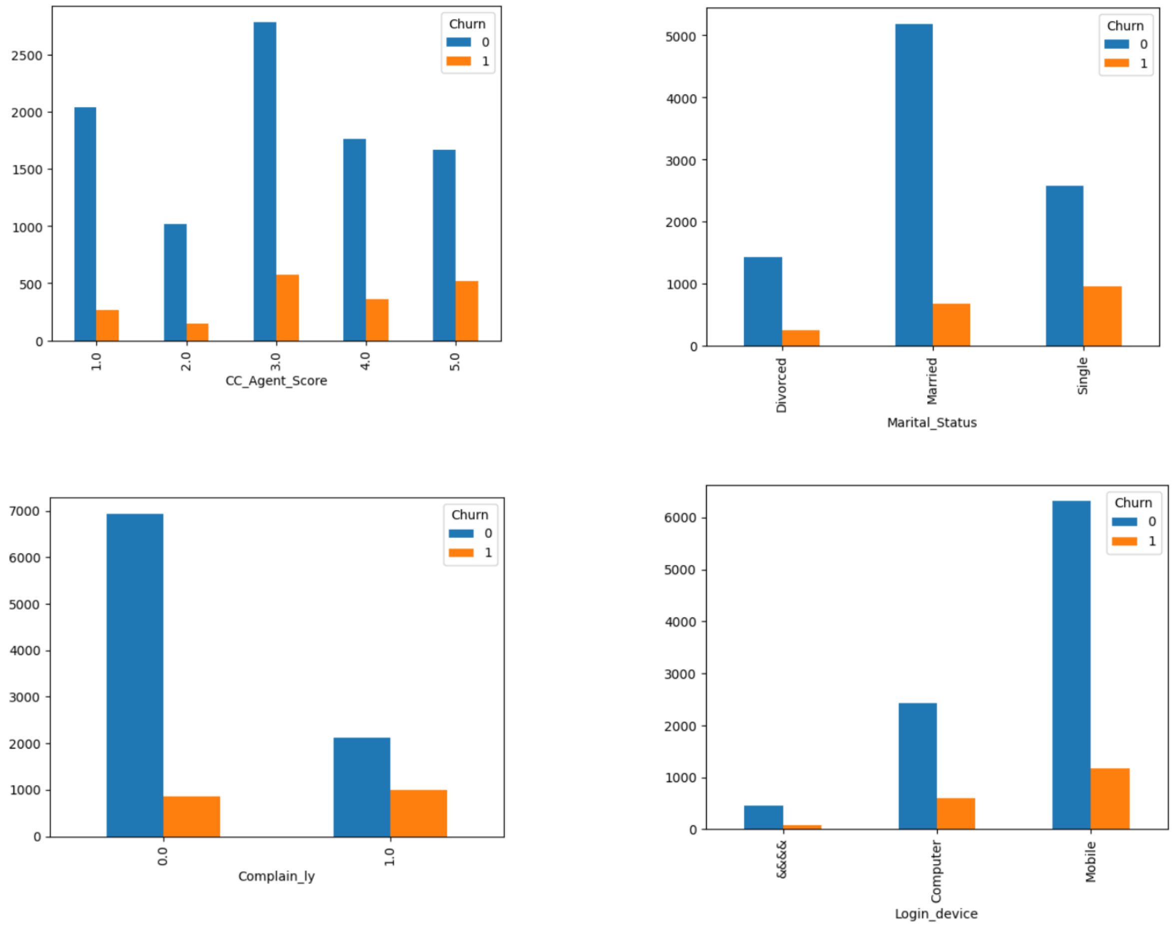


Fig 8: Bivariate Analysis of dataset

- Customers who prefer using "debit card" and "credit card" as their payment methods are more susceptible to churn
- Male customers demonstrate a higher churn ratio compared to their female counterparts
- City Tier 1 exhibits a higher churn rate compared to City Tiers 2 and 3
- Single customers are more inclined to churn compared to those who are divorced or married
- The "Regular Plus" segment has a higher churn rate among customers
- Customers accessing services via mobile devices are more likely to experience churn

Pairplot

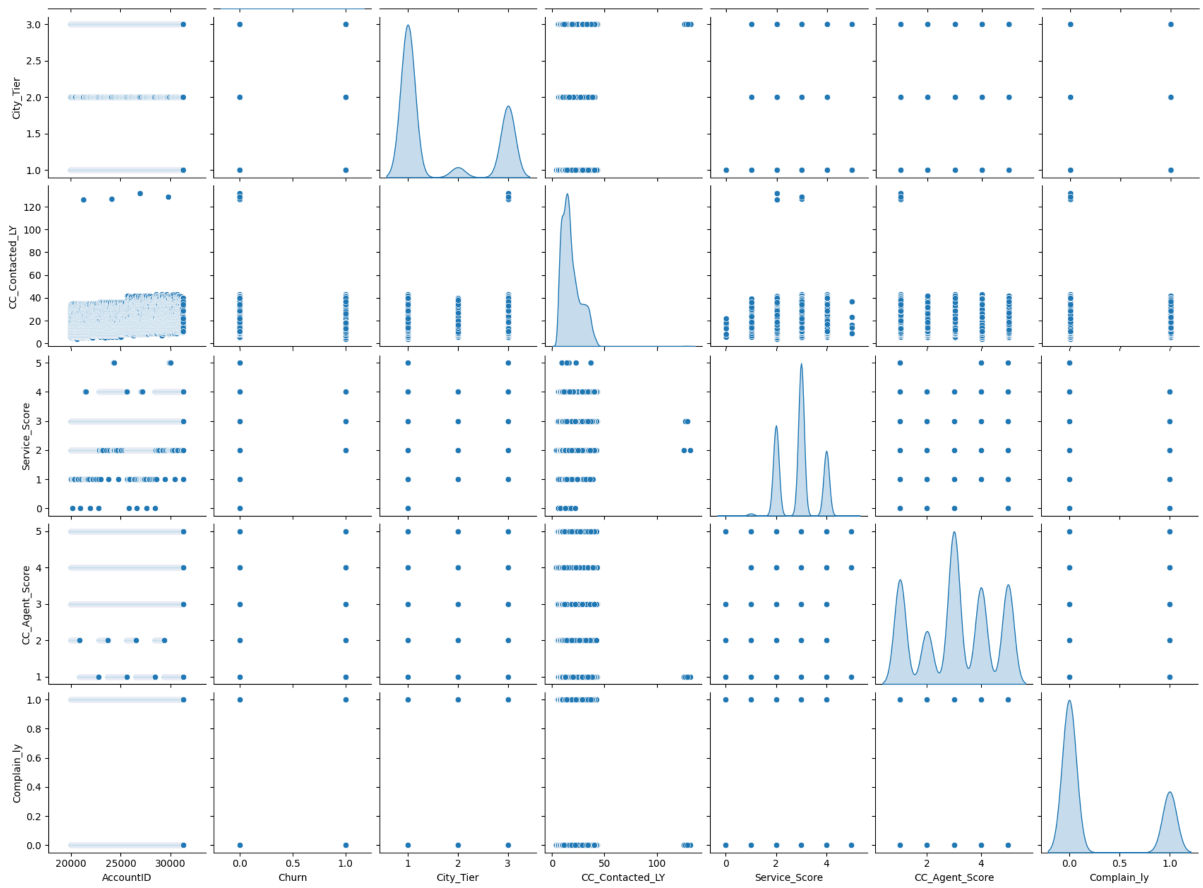


Fig 9: Pairplot of dataset

Heatmap:

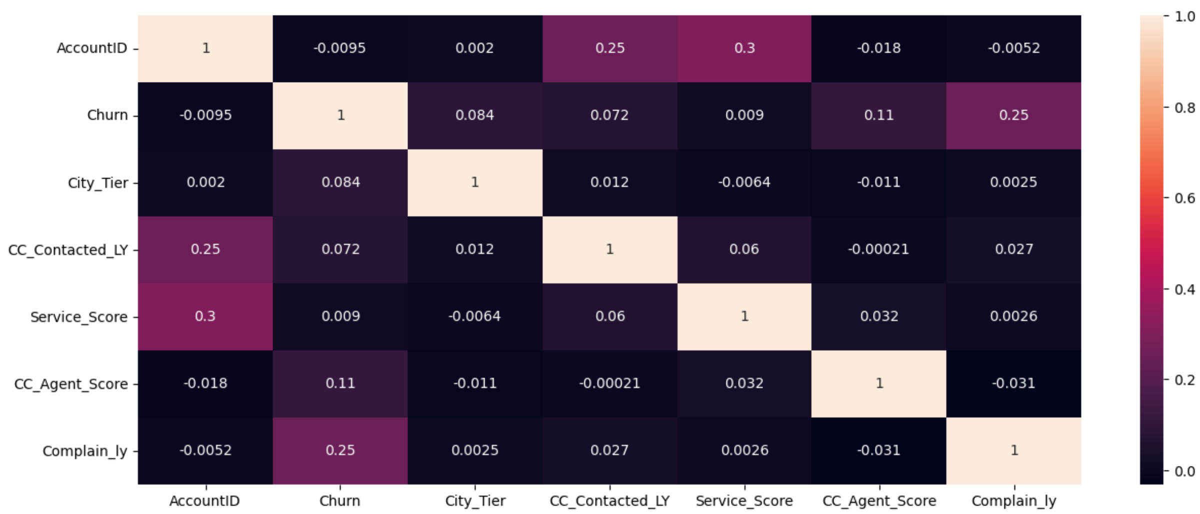


Fig 10: Heatmap of dataset

- A moderate positive correlation (0.25) exists between customer care contacts in the last year ('CC_Contacted_LY') and service scores ('Service_Score').
- There is a moderate positive correlation (0.25) between customer complaints in the last year ('Complain_ly') and the likelihood of churn ('Churn').

3. Data Cleaning and Pre-processing

Removal of unwanted variables (if applicable)

We can drop “Account ID” as it does not hold any significance in our analysis

New dataset:

	Churn	Tenure	City_Tier	CC_Contacted_LY	Payment	Gender	Service_Score	Account_user_count	account_segment	CC_Agent_Score	Marital_Status	rev_per_month	Complain_ly	rev_growth_yoy	coupon_used_for_payment	Day_Since_CC_connect	cashback	Login_device
0	1	4	3.0	6.0	Debit Card	Female	3.0	3	Super	2.0	Single	9	1.0	11	1	5	159.93	Mobile
1	1	0	1.0	8.0	UPI	Male	3.0	4	Regular Plus	3.0	Single	7	1.0	15	0	0	120.9	Mobile
2	1	0	1.0	30.0	Debit Card	Male	2.0	4	Regular Plus	3.0	Single	6	1.0	14	0	3	NaN	Mobile
3	1	0	3.0	15.0	Debit Card	Male	2.0	4	Super	5.0	Single	8	0.0	23	0	3	134.07	Mobile
4	1	0	1.0	12.0	Credit Card	Male	2.0	3	Regular Plus	5.0	Single	3	0.0	11	1	3	129.6	Mobile

Fig 11: Top 5 rows of new dataset

Missing Value treatment (if applicable)

Dataset has missing values:

```
AccountID                  0
Churn                      0
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
rev_growth_yoy             0
coupon_used_for_payment    0
Day_Since_CC_connect     357
cashback                 471
Login_device             221
dtype: int64
```

Fig 12: Missing values in dataset

- Also from our univariate and bivariate analysis, we found a lot of bad data, so we will process missing/null values and bad data together

We had converted bad data into null value and then imputed them with Median and Mode.

- **Tenure**: It had “#” as an anomaly, so replaced “#” with “nan” and further we replace “nan” with median
- **City_Tier**: It had null value, so we replaced them using mode
- **CC_Contacted_LY**: It had null value, so we replaced them using median
- **Payment**: It had null value, so we replaced them using mode
- **Gender**: It had “F” and “M” as anomaly, so replaced “F” with “Female” and “M” with “Male”. Further we replaced the “nan” using Mode
- **Service_Score**: It had null value, so we replaced them using mode
- **Account_user_count**: It had “@” as an anomaly, so replaced “@” with “nan” and further we replace “nan” with median
- **account_segment**: It had “Regular +” and “Super +” as anomaly, so replaced “Regular +” with “Regular Plus” and “Super +” with “Super Plus”. Further we replaced the “nan” using Mode
- **CC_Agent_Score**: It had null value, so we replaced them using mode
- **Marital_Status**: It had null value, so we replaced them using mode
- **rev_per_month**: It had “+” as an anomaly, so replaced “+” with “nan” and further we replace “nan” with median
- **Complain_ly**: It had null value, so we replaced them using mode
- **rev_growth_yoy**: It had “$” as an anomaly, so replaced “$” with “nan” and further we replace “nan” using median
- **coupon_used_for_payment**: It had “#”,”$”, and “*” as an anomaly, so replaced them with “nan” and further we replace “nan” using median
- **Day_Since_CC_connect**: It had “$” as an anomaly, so replaced “$” with “nan” and further we replace “nan” using median
- **Cashback**: It had “$” as an anomaly, so replaced “$” with “nan” and further we replace “nan” using median
- **Login_device**: It had “&&&&” as an anomaly, so replaced “&&&&” with “nan” and further we replace “nan” using mode

Dataset after Null value treatment:

```
Churn                      0
Tenure                     0
City_Tier                  0
CC_Contacted_LY            0
Payment                    0
Gender                     0
Service_Score              0
Account_user_count         0
account_segment            0
CC_Agent_Score             0
Marital_Status             0
rev_per_month              0
Complain_ly                0
rev_growth_yoy             0
coupon_used_for_payment    0
Day_Since_CC_connect       0
cashback                   0
Login_device               0
dtype: int64
```

Fig 13: Missing values after treatment

Outlier treatment

We converted all the data types into integer:

```
Churn                      int64
Tenure                     Int64
City_Tier                  Int64
CC_Contacted_LY            Int64
Payment                    int64
Gender                     int64
Service_Score              int64
Account_user_count         int64
account_segment            int64
CC_Agent_Score             int64
Marital_Status             int64
rev_per_month              int64
Complain_ly                int64
rev_growth_yoy             int64
coupon_used_for_payment    int64
Day_Since_CC_connect       int64
cashback                   int64
Login_device               int64
dtype: object
```

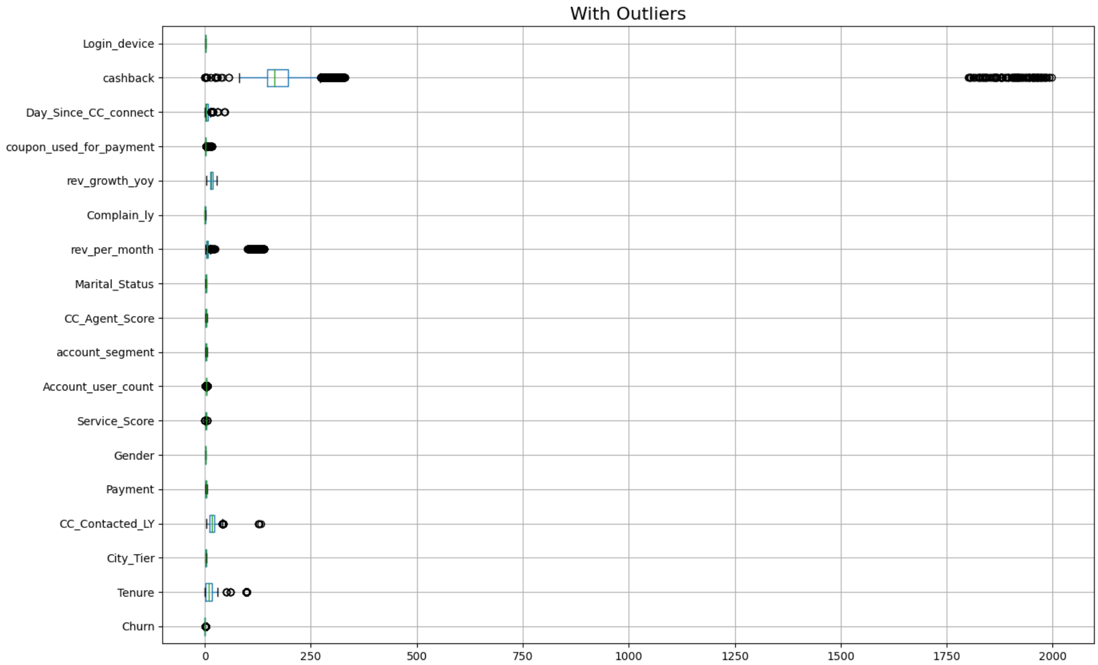


Fig 14: Outliers in Dataset

We used IQR method to treat outliers. Lower range and upper range for the attributes:

```
Lower Range for Tenure: -19.0
Upper Range for Tenure: 37.0
Lower Range for CC_Contacted_LY: -7.0
Upper Range for CC_Contacted_LY: 41.0
Lower Range for Account_user_count: 1.5
Upper Range for Account_user_count: 5.5
Lower Range for cashback: 72.0
Upper Range for cashback: 272.0
Lower Range for rev_per_month: -3.0
Upper Range for rev_per_month: 13.0
Lower Range for Day_Since_CC_connect: -5.5
Upper Range for Day_Since_CC_connect: 14.5
Lower Range for coupon_used_for_payment: -0.5
Upper Range for coupon_used_for_payment: 3.5
Lower Range for rev_growth_yoy: 4.0
Upper Range for rev_growth_yoy: 28.0
```

Fig 15: Lower and upper ranges in Dataset

After Outliers Treatment:

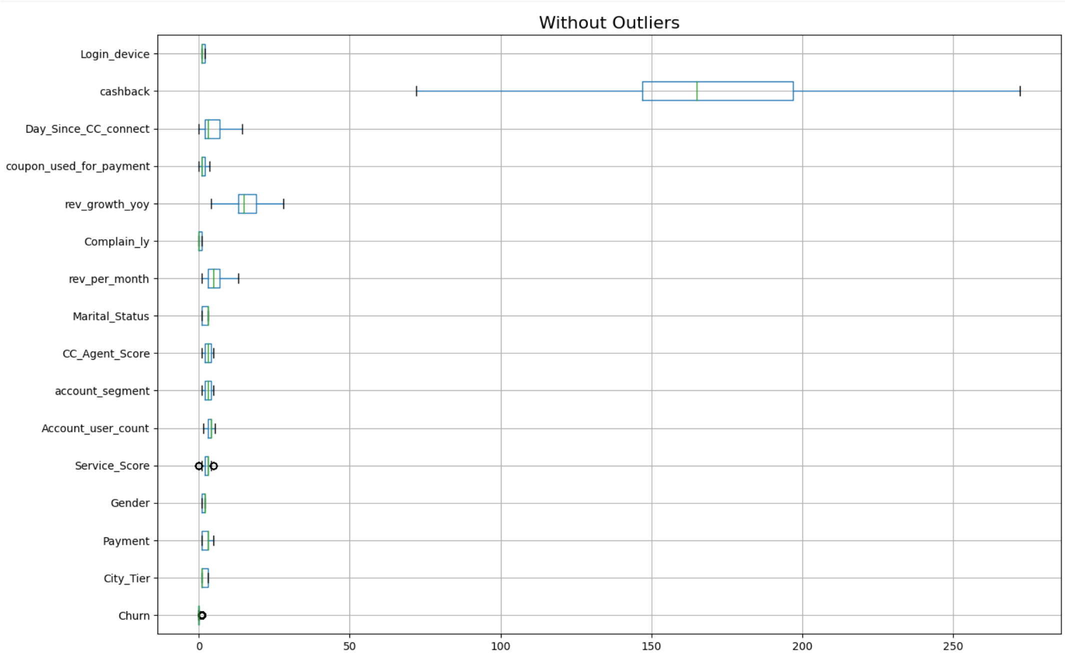


Fig 16: Outliers after treatment

Variable transformation (if applicable)

- Observing the dataset, it's evident that various variables possess distinct dimensions. For instance, "Cashback" represents currency, while "CC_Agent_Score" indicates a rating provided by customers. Consequently, these variables exhibit differing statistical characteristics.
- To address this disparity and ensure uniformity, scaling becomes imperative. By employing MinMaxScaler, we can normalize the data, bringing the standard deviation closer to zero. This normalization process is crucial for achieving consistency across diverse variables and facilitating more effective analysis.

Dataset after scaling:

	Churn	Tenure	City_Tier	CC_Contacted_LY	Payment	Gender	Service_Score	Account_user_count	account_segment	CC_Agent_Score	Marital_Status	rev_per_month	Complain_ly	rev_growth_yoy	coupon_used_for_payment	Day_Since_CC_connect	cashback	Login_device
0	1.0	0.108108	1.0	0.054054	0.00	0.0	0.625	0.375	0.75	0.25	0.0	0.666667	1.0	0.291667	0.285714	0.344828	0.435	0.0
1	1.0	0.000000	0.0	0.108108	0.25	1.0	0.625	0.625	0.25	0.50	0.0	0.500000	1.0	0.458333	0.000000	0.000000	0.240	0.0
2	1.0	0.000000	0.0	0.702703	0.00	1.0	0.375	0.625	0.25	0.50	0.0	0.416667	1.0	0.416667	0.000000	0.206897	0.465	0.0
3	1.0	0.000000	1.0	0.297297	0.00	1.0	0.375	0.625	0.75	1.00	0.0	0.583333	0.0	0.791667	0.000000	0.206897	0.310	0.0
4	1.0	0.000000	0.0	0.216216	0.50	1.0	0.375	0.375	0.25	1.00	0.0	0.166667	0.0	0.291667	0.285714	0.206897	0.285	0.0

Fig 17: Dataset after scaling

4. Model Building

From the above visual and non-visual analysis, we can say that it's a case of classification model, where in the target variable needs to be classified into "Yes" or "No". We will engage in model tuning and assess performance using various metrics, including Accuracy, F1 Score, Recall, Precision, ROC curve, AUC score, Confusion Matrix, and Classification Report. Our objective is to select a model that achieves optimal accuracy without exhibiting underfitting or overfitting.

Splitting Data into Train and Test Dataset:

We have split the data into 70:30 ratio and use this as the base for building models

We have assumed several supervised learning models

Logistic Regression - Logistic regression is a statistical method used for binary classification, predicting the probability of an event occurring. It models the relationship between the independent variables and the log-odds of the dependent variable using the logistic function. The output is transformed to a probability between 0 and 1, making it suitable for tasks like spam detection or medical diagnosis

Linear Discriminant Analysis (LDA) - Linear Discriminant Analysis (LDA) is a classification algorithm that finds the linear combination of features that best separates two or more classes. It maximizes the distance between class means while minimizing the spread within each class. LDA is commonly used for dimensionality reduction and is effective when the assumptions of normally distributed classes and equal covariance matrices are met

KNN - K-Nearest Neighbours (KNN) is a simple, instance-based learning algorithm used for classification and regression tasks. It assigns a data point's class label based on the majority class of its k nearest neighbours in the feature space. The choice of 'k' determines the level of complexity and smoothness in decision boundaries. KNN is non-parametric and versatile but can be sensitive to the scale of features.

Naïve Bayes - Naïve Bayes is a probabilistic machine learning algorithm based on Bayes' theorem. It assumes independence between features, hence the "naïve" designation. It is commonly used for classification tasks, particularly in natural language processing and spam filtering. Naïve Bayes calculates the probability of each class given a set of features and assigns the class with the highest probability to the input data.

Random Forest - Random Forest is an ensemble learning algorithm that builds multiple decision trees during training and merges their predictions for more robust and accurate results. It introduces randomness by training each tree on a random subset of the data and using a random subset of features at each split. Random Forest is effective for classification and regression tasks, offering improved generalization and resistance to overfitting.

Ada- Boosting - AdaBoost (Adaptive Boosting) is an ensemble learning method that combines multiple weak classifiers to create a strong classifier. It assigns weights to instances and focuses on misclassified ones in subsequent iterations to improve overall accuracy. AdaBoost is particularly effective in binary classification problems and is known for its ability to adapt to complex decision boundaries

Gradient Boosting - Gradient Boosting is an ensemble learning technique that builds a predictive model by combining the outputs of multiple weak learners, usually decision trees. It works by sequentially fitting new models to the residual errors of the previous models, optimizing for both bias and variance. Gradient Boosting is widely used for regression and classification tasks due to its ability to create powerful and accurate predictive models

The data is unbalanced, the count of dependent variable is 9364 for 0 and 1896 for 1.

We have employed SMOTE technique to recover from this imbalance.

After SMOTE:

```
X_train_res (13112, 17)
y_train_res (13112,)
```

Fig 18: Shape of Dataset after SMOTE

After building various models and analysing various parameters we conclude that "KNN" with default values out performs all other models built. We have concluded this basis Accuracy, F1 score, Recall, Precision and AUC score

KNN Model

The accuracy scores obtained from this model:

```
Accracy of training dataset: 0.8514336462826694
Accuracy for testing dataset 0.8342214328004737
```

Fig 19: Accuracy of the KNN Model

Confusion matrix obtained from this model:

```
confusion matrix of training dataset
[[6278  297]
 [ 874  433]]

confusion matrix for testing dataset
[[2634  155]
 [ 405  184]]
```

Fig 20: Confusion Matrix of the KNN Model

Classification Report obtained from this model:

```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       0.88      0.95      0.91      6575
           1       0.59      0.33      0.43      1307

    accuracy                           0.85      7882
   macro avg       0.74      0.64      0.67      7882
weighted avg       0.83      0.85      0.83      7882


Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.87      0.94      0.90      2789
           1       0.54      0.31      0.40       589

    accuracy                           0.83      3378
   macro avg       0.70      0.63      0.65      3378
weighted avg       0.81      0.83      0.82      3378
```

Fig 21: Classification Report of the KNN Model

AUC scores and ROC curves obtained from this model:

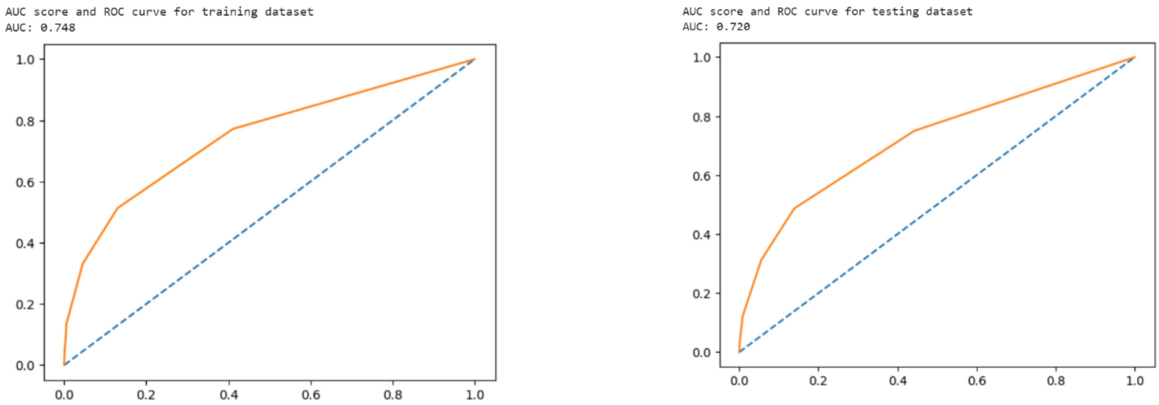


Fig 22: AUC scores and ROC curves of the KNN Model

The 10-fold cross validation scores:

```
cross validation scores for train dataset
array([0.83396705, 0.84664132, 0.82106599, 0.83121827, 0.83502538,
       0.83248731, 0.82741117, 0.82233503, 0.84137056, 0.82614213])
```

```
cross validation scores for test dataset
array([0.84023669, 0.80769231, 0.81952663, 0.81360947, 0.78994083,
       0.81952663, 0.80177515, 0.80177515, 0.80712166, 0.82492582])
```

Fig 23: 10-fold cross validation scores of the KNN

Finding the right value of n_neighbor:

Determining the optimal value for n_neighbors is crucial for achieving the highest accuracy in the model. The selection of the best n_neighbors can be guided by assessing Mean Squared Error (MSE) scores. The value that results in the lowest MSE signifies the least amount of error and, consequently, represents the most optimized choice for n_neighbors.

MSE scores:

```
[0.2066311426879811,
 0.18946121965660156,
 0.16577856719952633,
 0.16725873297809357,
 0.15926583777383063,
 0.15837773830669033,
 0.1604499703966844,
 0.16222616933096512,
 0.1613380698638247,
 0.16015393724097093]
```

Fig 24: MSE scores of the KNN Model

MSE Scores plot:

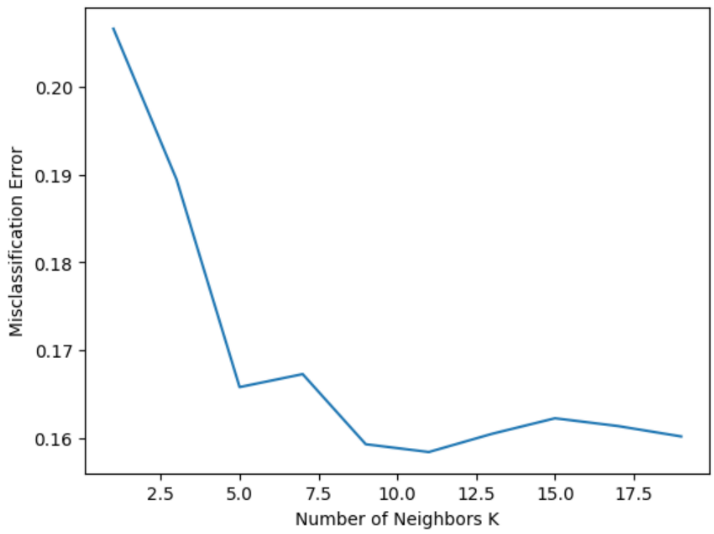


Fig 25: MSE scores plot of the KNN Model

- The graph depicted above illustrates that the value of "5" for n_neighbors yields the minimum MSE score. Therefore, we can proceed to construct the KNN model with n_neighbors set to "5," which coincidentally is the default value for n_neighbors. Consequently, there is no need for building different models with varying n_neighbor values, as the default value suffices in this case.

The accuracy scores obtained from this model (n=5):

```
Accuracy for training dataset: 0.8514336462826694
Accuracy score for testing dataset: 0.8342214328004737
```

Fig 26: Accuracy of the KNN Model (n=5)

Confusion matrix obtained from this model:

```
confusion matrix for training dataset
[[6278  297]
 [ 874  433]]

confusion matrix for testing dataset
[[2634  155]
 [ 405  184]]
```

Fig 27: Confusion matrix of the KNN Model (n=5)

Classification Report obtained from this model:

```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       0.88      0.95      0.91      6575
           1       0.59      0.33      0.43      1307

    accuracy                           0.85      7882
   macro avg       0.74      0.64      0.67      7882
weighted avg       0.83      0.85      0.83      7882


Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.87      0.94      0.90      2789
           1       0.54      0.31      0.40       589

    accuracy                           0.83      3378
   macro avg       0.70      0.63      0.65      3378
weighted avg       0.81      0.83      0.82      3378
```

Fig 28: Classification Report of the KNN Model (n=5)

AUC scores and ROC curves obtained from this model:

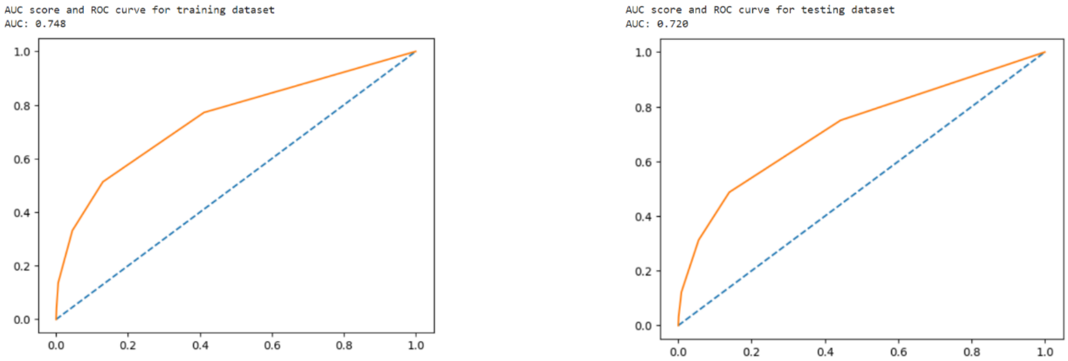


Fig 29: AUC scores and ROC curves of the KNN Model (n=5)

The 10-fold cross validation scores:

```
cross validation scores for training dataset
array([0.83396705, 0.84664132, 0.82106599, 0.83121827, 0.83502538,
       0.83248731, 0.82741117, 0.82233503, 0.84137056, 0.82614213])
```

```
cross validation scores for testing dataset
array([0.84023669, 0.80769231, 0.81952663, 0.81360947, 0.78994083,
       0.81952663, 0.80177515, 0.80177515, 0.80712166, 0.82492582])
```

Fig 30: 10-fold cross validation scores of the KNN Model

KNN model using GridSearchCV

The accuracy scores obtained from this model:

```
Accuracy of training dataset after gridsearchCV: 0.8613296117736615
Accuracy of testing dataset after gridsearchCV: 0.8436944937833037
```

Fig 31: Accuracy scores of the KNN Model (CV)

Confusion matrix obtained from this model:

```
confusuon matrix for training dataset
array([[6390,  185],
       [ 908,  399]])
```

```
confusuon matrix for testing dataset
array([[2687,  102],
       [ 426,  163]])
```

Fig 32: Confusion Matrix of the KNN Model (CV)

Classification Report obtained from this model:

```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       0.88      0.97      0.92      6575
           1       0.68      0.31      0.42      1307

    accuracy                           0.86      7882
   macro avg       0.78      0.64      0.67      7882
weighted avg       0.84      0.86      0.84      7882


Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.86      0.96      0.91      2789
           1       0.62      0.28      0.38       589

    accuracy                           0.84      3378
   macro avg       0.74      0.62      0.65      3378
weighted avg       0.82      0.84      0.82      3378
```

Fig 33: Classification Report of the KNN Model (CV)

AUC scores and ROC curves obtained from this model:

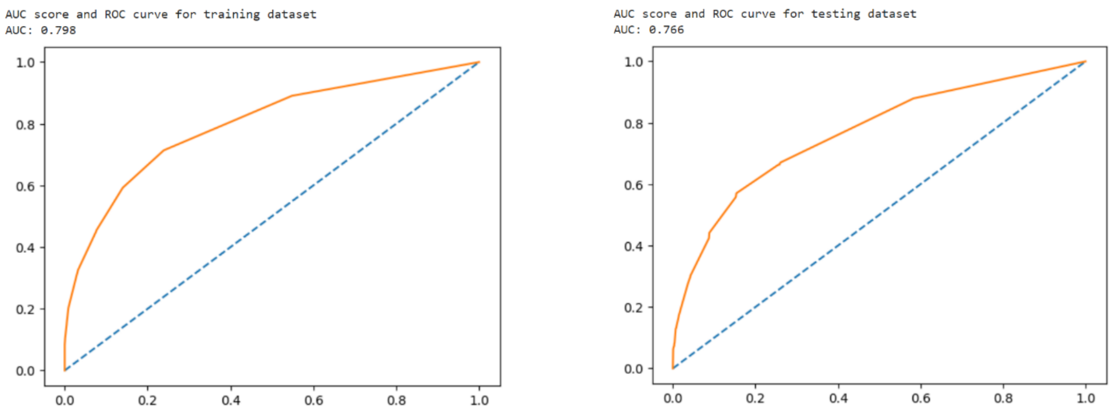


Fig 34: AUC scores and ROC curves of the KNN Model (CV)

The 10-fold cross validation scores:

```
cross validation scores for train dataset
array([0.83776933, 0.84157161, 0.83502538, 0.84517766, 0.84010152,
       0.84137056, 0.85786802, 0.85532995, 0.83629442, 0.83502538])


cross validation scores for test dataset
array([0.84023669, 0.83431953, 0.82248521, 0.83431953, 0.82840237,
       0.84615385, 0.83431953, 0.80473373, 0.82492582, 0.83086053])
```

Fig 35: 10-fold cross validation scores of the KNN Model (CV)

KNN model using SMOTE

The accuracy scores obtained from this model:

```
Accuracy of training dataset: 0.7321673003802281
Accuracy of testing dataset: 0.7045589105979869
```

Fig 36: Accuracy scores of the KNN Model (SMOTE)

Confusion matrix obtained from this model:

```
confusion matrix for training dataset
array([[4781, 1794],
       [1728, 4847]])


confusion matrix for testing dataset
array([[1975,  814],
       [ 184,  405]])
```

Fig 37: Confusion matrix of the KNN Model (SMOTE)

Classification Report obtained from this model:

```
Classification Report of the training data:

              precision    recall  f1-score   support

           0       0.73      0.73      0.73      6575
           1       0.73      0.74      0.73      6575

    accuracy                           0.73     13150
   macro avg       0.73      0.73      0.73     13150
weighted avg       0.73      0.73      0.73     13150


Classification Report of the test data:

              precision    recall  f1-score   support

           0       0.91      0.71      0.80      2789
           1       0.33      0.69      0.45       589

    accuracy                           0.70      3378
   macro avg       0.62      0.70      0.62      3378
weighted avg       0.81      0.70      0.74      3378
```

Fig 38: Classification Report of the KNN Model (SMOTE)

AUC scores and ROC curves obtained from this model:

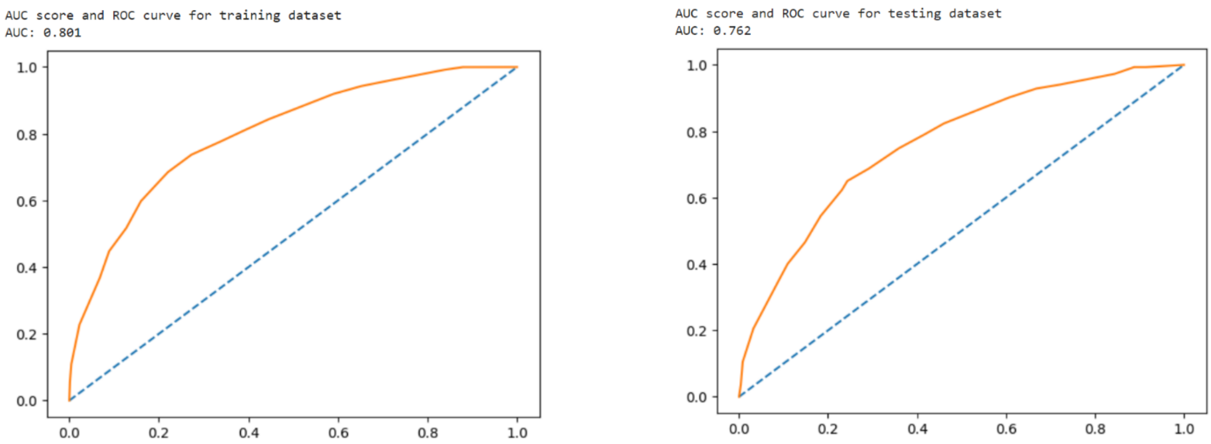


Fig 39: AUC scores and ROC curves of the KNN Model (SMOTE)

The 10-fold cross validation scores:

```
cross validation scores for train dataset
array([0.69581749, 0.73536122, 0.72319392, 0.69961977, 0.70190114,
       0.71406844, 0.73688213, 0.71787072, 0.69277567, 0.72091255])

cross validation scores for test dataset
array([0.83727811, 0.83431953, 0.83727811, 0.82840237, 0.83727811,
       0.83136095, 0.82544379, 0.81952663, 0.83382789, 0.82195846])
```

Fig 40: 10-fold cross validation scores of the KNN Model (SMOTE)

Inferences with KNN

Based on the observations, it can be deduced that the data exhibits neither "Overfitting" nor "Underfitting" characteristics. The model created using GridSearchCV is deemed to be the most optimized, given the obtained best parameters. However, it is noteworthy that the accuracy scores, as well as recall, precision, F1 values, ROC curve, and AUC score, do not show a substantial improvement when compared to models built with default values and SMOTE dataset

Moreover, the model constructed on the SMOTE dataset have a noticeable decline in accuracy in both training and testing dataset

Inference from final model

- After evaluating scores across various models for both the training and testing datasets, it is evident that the KNN model with default hyperparameter values is the most optimized for the provided dataset.
- The accuracy difference between Logistic Regression and Linear Discriminant Analysis (LDA) is minimal, with LDA showcasing slightly superior performance.
- Models employing bagging and boosting techniques are well-optimized; however, there is a slightly higher accuracy gap between the training and testing datasets compared to KNN.
- Naïve Bayes and LDA models demonstrated strong performance on the training dataset but exhibited reduced accuracy when applied to the testing dataset, suggesting overfitting.
- All models constructed on balanced datasets displayed signs of overfitting

All Model comparison

	Train Data								Test Data							
	Accuracy	Precision		Recall		F1-Score		AUC Score	Accuracy	Precision		Recall		F1-Score		AUC Score
		0	1	0	1	0	1			0	1	0	1	0	1	
Logistic Regression Model	84%	0.85	0.61	0.99	0.12	0.91	0.2	0.75	83%	0.84	0.62	0.98	0.12	0.91	0.2	0.75
Logistic Regression model (CV)	84.1%	0.85	0.61	0.99	0.12	0.91	0.2	0.75	83.3%	0.84	0.62	0.98	0.12	0.91	0.2	0.749
Logistic Regression model (SMOTE)	67.6%	0.68	0.68	0.68	0.68	0.68	0.68	0.755	67.6%	0.91	0.31	0.67	0.69	0.77	0.43	0.742
LDA Model	84.1%	0.85	0.59	0.98	0.16	0.91	0.25	0.742	83.4%	0.85	0.59	0.98	0.17	0.91	0.26	0.742
LDA Model (CV)	84.1%	0.85	0.59	0.98	0.16	0.91	0.25	0.747	83.3%	0.85	0.58	0.97	0.17	0.91	0.26	0.746
LDA Model (SMOTE)	67.9%	0.68	0.68	0.68	0.68	0.68	0.68	0.754	67.8%	0.91	0.31	0.67	0.70	0.78	0.43	0.744
KNN Model	85.1%	0.88	0.59	0.95	0.33	0.91	0.43	0.748	83.4%	0.87	0.54	0.94	0.31	0.90	0.40	0.720
KNN Model (n=5)	85.1%	0.88	0.59	0.95	0.33	0.91	0.43	0.748	83.4%	0.87	0.54	0.94	0.31	0.90	0.40	0.720
KNN Model (CV)	86.1%	0.88	0.68	0.97	0.31	0.92	0.42	0.798	84.3%	0.86	0.62	0.96	0.28	0.91	0.38	0.766
KNN Model (SMOTE)	73.2%	0.73	0.73	0.73	0.74	0.73	0.73	0.801	70.4%	0.91	0.33	0.71	0.69	0.80	0.45	0.762
Naïve Bayes Model	28.9%	0.93	0.18	0.16	0.94	0.27	0.31	0.719	28.3%	0.91	0.19	0.15	0.93	0.25	0.31	0.705
Naïve Bayes Model (SMOTE)	56.2%	0.80	0.54	0.17	0.96	0.28	0.69	0.730	28.9%	0.91	0.19	0.16	0.92	0.27	0.31	0.706
Random Forest	86.2%	0.87	0.72	0.98	0.28	0.92	0.40	0.837	84.3%	0.86	0.64	0.97	0.24	0.91	0.35	0.795
Random Forest (SMOTE)	74.7%	0.74	0.76	0.77	0.73	0.75	0.74	0.831	73.0%	0.92	0.36	0.74	0.68	0.82	0.47	0.785
Bagging	86.1%	0.87	0.72	0.98	0.27	0.92	0.39	0.834	84.4%	0.86	0.64	0.97	0.24	0.91	0.35	0.788
Bagging (SMOTE)	74.7%	0.75	0.75	0.75	0.74	0.75	0.75	0.833	72.2%	0.92	0.35	0.73	0.69	0.81	0.46	0.781
Ada-Boost Model	83.9%	0.85	0.59	0.99	0.10	0.91	0.17	0.751	83.1%	0.84	0.59	0.98	0.10	0.91	0.18	0.748
Ada-Boost Model (SMOTE)	67.9%	0.68	0.68	0.68	0.68	0.68	0.68	0.753	67.5%	0.91	0.31	0.67	0.69	0.77	0.43	0.741
Gradient Boosting	84.7%	0.85	0.66	0.98	0.16	0.91	0.26	0.780	83.8%	0.85	0.65	0.98	0.16	0.91	0.25	0.770
Gradient Boosting (SMOTE)	71.0%	0.69	0.73	0.75	0.67	0.72	0.70	0.784	72.3%	0.91	0.35	0.74	0.65	0.82	0.45	0.760

Fig 41: All Model comparison table

5. Model Validation

Model validation involves assessing the performance and generalization ability of a machine learning model. Accuracy is just one metric used for validation; other metrics provide a more comprehensive evaluation. Common validation metrics include precision, recall, F1 score, and area under the receiver operating characteristic curve (AUC-ROC) for classification tasks.

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.

$$\text{Accuracy} = (TP + TN) / (TP + FP + TN + FN)$$

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

$$\text{Precision} = TP / (TP + FP)$$

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

$$\text{Recall} = TP/ (TN + FN)$$

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

$$\text{Specificity} = TN/ (TN + FP)$$

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.

6. Final interpretation / recommendation

Insights from data

- Primary business visibility in Tier-1 cities
- Predominantly received customer service ratings of "3"
- Customer care interactions typically rated as "3"
- Low transaction frequency via UPI and e-wallet
- Higher churn observed in the "Regular+" account segment
- Maximum churn associated with customers having a "single" marital status
- No apparent correlation between complaints raised in the last 12 months and churn
- Direct proportional relationship between customer tenure and cashback
- Tier-wise computer usage: Tier 1 > Tier 3 > Tier 2 cities

The data collection reveals significant variations, encompassing a mix of services, customer ratings, and profiles. Here are key insights and recommendations:

- **Expansion in Tier 2 Cities**: The business should enhance its presence in tier 2 cities to increase visibility and attract new customers.
- **Promotion of Hassle-Free Payment Methods**: Encourage customers to adopt hassle-free and secure payment methods such as standing instructions in bank accounts or UPI.

- **Service Improvement through Customer Surveys**: Recognizing the need for improvement in service scores, the business should conduct surveys to gain a deeper understanding of customer expectations.
- **Customer Care Excellence**: Provide training for customer care executives to enhance customer experience, leading to improved feedback scores.
- **Curated Plans Based on Tenure**: Develop personalized plans not only based on customer spending but also considering their tenure with the business.
- **Tailored Plans for Families**: Introduce curated plans, like a family floater, specifically designed for married individuals to cater to their unique needs.

These strategies aim to address the diverse data insights and provide actionable recommendations for enhancing the business's performance and customer satisfaction.

Recommendations from Model building

- Leveraging the constructed models, businesses can devise effective strategies to enhance customer retention
- Implementing family floater offers and discounts can be a viable approach to incentivize customers to remain loyal
- Introducing regular discount coupons through the business's e-wallet platform could serve as an enticing incentive for customers
- Business needs to increase in visibility in Tier-2 city for better customer acquisition
- Providing discount vouchers for other vendors or future bills, contingent on meeting a minimum bill criteria, is another strategic option
- The models offer valuable insights into the current standing of the business, enabling informed decisions on how to improve and optimize customer retention strategies

Thank you