MBA Semester – IV

Capstone Project – Final Report

Name	Deepak TM, J K Ramakrishna Chitrada, Adarsh S Shetty, C Suraj, Ambalakara Indusree
Project	Customer Churn
Group	Group 7 - CCP
Date of Submission	19/11/2023

A study to predict Customer Churn Problems in an Organization Using EDA and Machine Learning Models using Python

Capstone Project submitted to Jain Online (Deemed-to-be University)

In partial fulfilment of the requirements for the award of:

Master of Business Administration

Submitted by:

Name	USN
Deepak TM	212VMBR01912
J K Ramakrishna Chitrada	212VMBR01966
Adarsh S Shetty	212VMBR01794
C Suraj	212VMBR01888
Ambalakara Indusree	212VMBR01811

Under the guidance of

HRUSHIKESHA SHASTRY B S

(Faculty-JAIN Online)

Jain Online (Deemed-to-be University)

Bangalore

2022-23

DECLARATION

We, the below mentioned students, hereby declare that the Capstone Project Report titled **"A study to predict Customer Churn Problem in an Organization Using EDA and Machine Learning Models using Python"** has been prepared by us under the guidance of Mr. **Hrushikesha Shastry B S**. We declare that this Project work is towards the partial fulfillment of the University Regulations for the award of the degree of Master of Business Administration by Jain University, Bengaluru. We have undergone a project for Eight Weeks. We further declare that this Project is based on our original study and has not been submitted for the award of any degree/diploma from any other University / Institution.

Place: Bangalore

Date: 19/11/2023

Deepak TM (USN: 212VMBR01912),
J K Ramakrishna Chitrada (USN: 212VMBR01966),
Adarsh S Shetty (USN: 212VMBR01794),
C Suraj (USN: 212VMBR01888),
Ambalakara Indusree (USN: 212VMBR01811)

ACKNOWLEDGEMENT

As part of the MBA (Data Science and Analytics) Degree the students of Jain Online Master Degree (4th Semester) must submit a Capstone Project by incorporating the knowledge and technologies learned throughout the course. The Project work is about the Customer churn problem of a DTH company, and its objective is to identify the customer churn patterns and prepare an ML model for the company to identify the problem and make necessary recommendations. As a part of the project work various ML model works have been identified and tested for better output.

We would like to express our sincere thanks and gratitude to Jain Online University and its faculty. We would like to thank Mr. **Hrushikesha Shastry B S** for his mentorship, support and valuable input during the entire tenure of this project work.

We would like to acknowledge that the above work being submitted is part of own work of our academic team mentioned on the cover page of the project and is not copied. The project is purely for educational purposes and is NOT for any commercial use. Whenever external sources are used as input in this project work such sources are mentioned in the reference section of the project.

Executive Summary

Enhancing Customer Retention Strategies for Sustainable Business Growth

This project focuses on optimizing customer retention strategies to drive sustainable business growth by minimizing customer churn. Customer Churn is a critical concern for any business, as it directly impacts revenue, profitability, and overall organizational stability. This project aims to develop and implement a comprehensive customer retention strategy that maximizes customer engagement, satisfaction, and loyalty, ultimately reducing churn rates and increasing customer lifetime value.

The research and analysis conducted as part of the project indicate that understanding customer behaviour, preferences, and pain points is fundamental to devising effective retention strategies. Leveraging advanced analytics and artificial intelligence, we will analyze customer data to identify patterns and potential indicators of churn. This will enable us to proactively address customer concerns and tailor personalized approaches to retain at-risk customers.

The project proposes a customer churn prediction model that helps to devise strategies to retain customers by minimizing churn. The solution includes:

- **Data Collection and Exploratory Data Analysis**: Data collection refers to the collection of relevant data from appropriate sources for customer churn prediction. After collecting the data, Exploratory Data Analysis is done to identify the variables and features contributing to the customer churn. It also helps to identify the pattern and trends of each feature in the dataset and its importance in the business concept. It also helps to uncover the business problem.

- **Churn prediction model development**: The project employs different machine learning algorithms such as Logistic Regression, Linear Discriminant Analysis, Random Forest, K-Nearest Neighbors, Support Vector Machine and XGBoost Classifier to build a robust prediction model. It also includes the performance evaluation of models to ensure the accuracy and reliability of the prediction.

- **Solution implementation**: The project focuses on the implementation of the best prediction model and its enhancement to develop clustering solutions for the segmentation of customers. This will enable the organization to strategize and target the high-risk, medium risk and low-risk customer segments.

TABLE OF CONTENTS

List of Tables

List of Graphs

CHAPTER 1

INTRODUCTION AND BACKGROUND

Abstract

Nowadays due to the rapid growth of human needs along with technology, the interests and preferences of the customers are also changing at a rapid pace. With the rise in competition across various industries, customer churn has emerged as a significant challenge for existing market players that need to be addressed. Customer churn creates a significant impact in the e-commerce, Telecom, DTH Subscriptions, DTH service domain etc. Losing customer accounts means a fall in revenue and profits. If the problem is not addressed in time it may lead to losses in the long run due to associated fixed costs and future competitors.

The problem statement states the urgent requirement for a predictive model that can identify potential churners. Leveraging the key features in the historical data the machine learning model classifies the customers who are most likely to churn. This includes account-specific attributes such as tenure, account segmentation, payment preferences, city tier, gender, service satisfaction scores and more.

The outcome of the project not only helps the organizations in mitigating the situation and fighting back against the customer churn. It also helps to build the recommendation model to proactively recommend solutions to devise strategies to target the segments and create effective campaigns.

Business Problem

A Direct-to-Home (DTH) streaming service that has been a prominent player in the entertainment industry for the past decade. The company was established to provide a diverse array of streaming content, including movies, TV shows, original series, documentaries, and more, accessible through various devices like smartphones, smart TVs, tablets, and computers.

However, in recent times, the DTH Service Provider has been facing significant challenges, primarily an alarming increase in customer churn rate. The Churn Rate refers to the percentage of subscribers who cancel their subscriptions during a specific period. This issue has been a cause of concern for the company as it directly impacts its revenue, growth prospects, and market share.

Problem Statement

In the competitive landscape of the DTH industry, acquiring and retaining customers is a significant challenge and a key business goal. The Customer Engagement team is committed to re-engaging users who have terminated their subscriptions in a bid to minimize churn rates. Currently, they use a traditional analysis approach using Excel and develop strategies according to analysis which require more human intervention and can be erroneous. They propose an AI-integrated prediction model that can automate and eliminate maximum human intervention in terms of data analysis and churn prediction. They wanted to use the model to predict the potential churners to develop customer retention strategies and campaigns. They have collected historical records and provided for the project. The dataset collected by the company has both the metadata in the first tab and historical records in the second tab.

Table 1: Attribute name and its description

Variable	Description
AccountID	account unique identifier
Churn	account churn flag (Target)
Tenure	Tenure of account
City_Tier	Tier of primary customer's city
CC_Contacted_L12m	How many times all the customers of the account have contacted customer care in the last 12 months
Payment	Preferred Payment mode of the customers in the account
Gender	Gender of the primary customer of the account
Service_Score	Satisfaction score given by customers of the account on service provided by the company
Account_user_count	Number of customers tagged with this account
account_segment	Account segmentation based on spend
CC_Agent_Score	Satisfaction score given by customers of the account on customer care service provided by the company
Marital_Status	Marital status of the primary customer of the account
rev_per_month	Monthly average revenue generated by account in last 12 months
Complain_l12m	Any complaints have been raised by the account in the last 12 months
rev_growth_yoy	revenue growth percentage of the account (last 12 months vs last 13 to 24 months)
coupon_used_l12m	How many times customers have used coupons to make the payment in the last 12 months

Day_Since_CC_connect	Number of days since no customers in the account have contacted customer care
cashback_l12m	Monthly average cashback generated by account in last 12 months
Login_device	Preferred login device of the customers in the account

Objective of the study and need for a project

Business objective

By identifying patterns and trends in the data, companies can easily focus on the features inclined to customer churn problems. The organization can act quickly based on the insight from the identified patterns and trends.

By building a machine learning model the DTH Service Provider can easily target the probable churners and devise strategies to retain them by providing offers, targeting personalized campaigns, and conducting market study.

Demand and Need of the Project

The customer is the king who has the power of purchase and choice. Companies compete to add new features to their products to retain and satisfy their customers. Novel technologies and advertisements can entice the customer to purchase their products. The never-ending wants and the desire to acquire the latest technology can lead to customer churn. So, it is inevitable for organizations to have a project to identify and study the pattern of losing customers and predict the possibility of future customer churns. It helps them to formulate strategies to attract and retain their valuable customers. It helps them to add value to their products. So, the demand for a project has become a necessity rather than a need.

Industry Overview

The Telecommunication, Broadband & DTH Industry is one of the rapidly growing and highly sensitive to technological changes. With the evolution of the higher spectrums like 4G and 5G in the Telecommunication industry all the other services in the industry are rapidly changing with the integrated technologies. Telecommunication service providers are advancing into Broadband Services and DTH services by making them into Bundle Packages. With this extensive advancement, the existing market players who exclusively provide Broadband or DTH services are facing severe competition both in terms of Pricing and Technologies. Due to this major customer

base is shifting from one service provider to other service providers or the integrated service providers. The problem statement of Customer Churn belongs to one such company.

Overview of the Theoretical Concepts

The Project Mainly involves the following concepts:

1. Exploratory Data Analysis
2. Machine Learning Model building
3. Model Evaluation
4. Model Optimization using Hyperparameter Tuning
5. Model Rebuilding and prediction

CHAPTER 2

RESEARCH METHODOLOGY

Scope of the Study

The ultimate objective of the study is to deploy a machine learning model to swiftly predict the probability of shedding customers and give light to the organization to build strategies to retain them. Identifying the features and their relationship with other variables is one of the scopes of the study. The study also includes the use of supervised learning models to identify the best model by evaluating its performance and implementing it.

Methodology

As discussed in earlier chapters, to predict the potential churners, the data needs to be collected, cleaned and used for identifying patterns, trends and their relationship with other features. Each step is a part of data understanding and business understanding. Once the data is analyzed, based on the insights Feature engineering is done to remove irrelevant features, add features if necessary and fix the anomalies and inconsistencies in the data. So, the methodology includes data collection, feature engineering, machine learning modelling, prediction and prediction evaluation, identification of best-performing model and implementation.

- **Research Design**

  The motto of research design is to guide the team through the entire research project. The research includes intense data analysis hovering through the length and breadth of the data. The entire project drives through the different phases of the "Cross-Industry Standard Process for Data Mining (CRISP-DM) Framework". While the project progresses through each phase of CRISP-DM it may come across obstacles such as inconsistent data, missing values, bad model performance etc. In this context, the project returns to the previous phases, fixes and resolves and moves toward deployment.

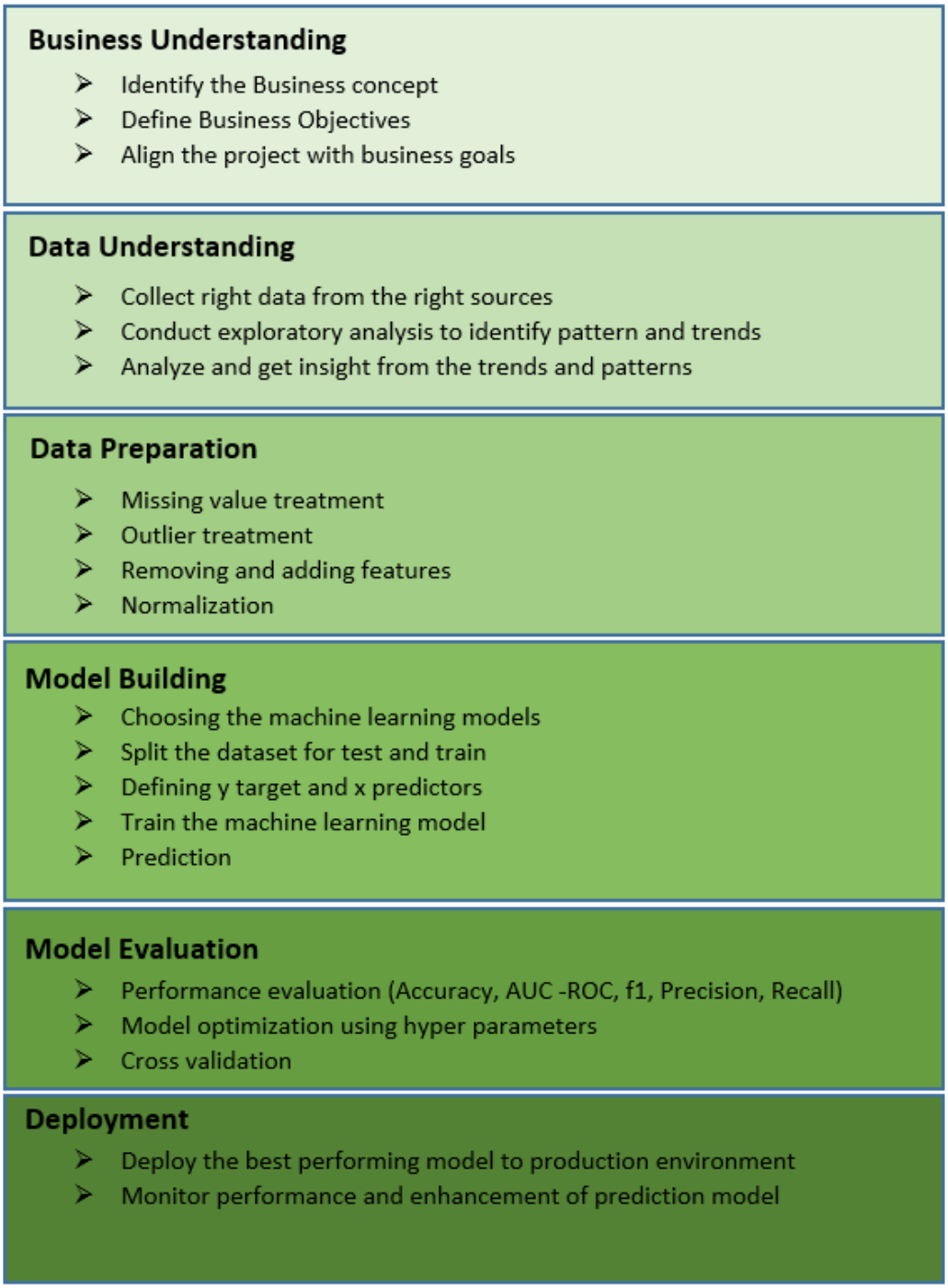

Figure 1: CRISP-DM

- **Data Collection**

  The data has already been collected and an initial review has been completed, identifying the dimension of the data, its data types and the presence of missing data.

- **Data Analysis Tools**

  There are several tools available for data analysis, but in this project, we have used Python for programming, Power BI and Excel for initial analysis. Although Tableau is very flexible in data analysis, it was not feasible as there were plenty of tools like Python, R, Power BI and Excel. Excel is not preferable for large datasets, but it is a robust tool for small datasets. Using Python, numerous libraries are used, some of them are Pandas, NumPy, sklearn, Matplotlib, and Seaborn.

  - Pandas: used for data manipulation.
  - NumPy: used for 2D array operations.
  - sklearn: used for ML model building and operations
  - Matplotlib, seaborn: used for plotting graphs.

There are plenty of machine learning algorithms available today, and some of the familiar machine learning models are chosen such as K-Nearest Neighbors, Logistic Regression, Support Vector Machine, Linear Discriminant Analysis, Extreme Gradient Boosting and Random Forest.

- The KNN model is a non-parametric algorithm that makes predictions based on the majority class among the k-nearest neighbors of a data point. It classifies the data points using Euclidean distance metrics by default to calculate the distance between data points.

- Linear Discriminant Analysis (LDA) is a supervised statistical technique used for dimensionality reduction and classification. It aims to reduce the dimensionality of data while maintaining class separability. LDA requires labelled data and is ideal for scenarios where you need to distinguish between different classes.

- Support Vector Machine is a versatile machine-learning tool used for classification and regression. It classifies the data identifying a hyperplane between two classes of data points.

The term support vectors refer to the data points lying close to the hyperplane which is crucial for defining the margin and the objective of the support vector machine is to maximize the distance between the hyperplane and the nearest data point of each class.

- Random Forest is one of the popular machine-learning models used for regression and classification. It classifies by splitting the entire dataset into different subsets and constructing decision trees for each subset. The output of each decision tree is subjected to majority voting in terms of classification and averaging in case of regression.
- Extreme gradient boosting (XGBoost) is a popular machine learning model that uses ensemble learning models in predicting. Similar to Random Forest it uses decision trees and boosting. It builds trees sequentially and each updates the residual errors by learning from its predecessors called boosting. It also uses a gradient descent function to minimize the loss function in each build.

Finally, the project will choose the best model based on its performance and accuracy. A Hyperparameter tuning will be done to ensure consistency and to optimise the performance.

Period of Study

The project used 8 years of data for the study and some of the features like Tenure of the account, the customer contacted in the last 12 months (CC_Contacted_L12m), revenue per month (rev_per_month) complaints raised by the account in the last 12 months (Complain_l12m) purely depends on time.

Utility of Research

The research enables the team to build a robust model customer churn-predicting solution that can meet the volume loads. It can be implemented in any e-commerce, telecom and DTH industry field. As the demand is very high, so is the utility of research.

CHAPTER 3

DATA ANALYSIS AND INTERPRETATION

Data Collection

The data collection for this study on predicting customer churn spanned about 99 months has been collected by the organization and the same has been received in a Excel file with two sheets in it one with attribute description and the other with the actual customer data. This time frame was carefully selected to ensure a comprehensive understanding of long-term customer engagement and potential churn patterns. The specific data contains information and activities for each month, including details such as product usage, customer service interactions, subscription renewals, complaints, etc.

Even though the description was given by the organization, a deep dig was required to understand each feature and to get more data insight. We used the Dataframe.head() function to identify quantitative and qualitative features. It is given in Understanding the Data reference code (DU3) in the Jupyter Notebook.

A glance of the data (DU3)

```
[ ] df.head(3)
```

tacted_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
6.0	Debit Card	Female	3.0	3	Super	2.0	Single	9	1.0	11	1	5	159.93	Mobile
8.0	UPI	Male	3.0	4	Regular Plus	3.0	Single	7	1.0	15	0	0	120.9	Mobile
30.0	Debit Card	Male	2.0	4	Regular Plus	3.0	Single	6	1.0	14	0	3	NaN	Mobile

Figure 2: First glance of data

While comparing the data against the data types in “Understanding the Data reference code (DU2)”, some numerical fields such as Tenure, Account_user_count, Revenue per month, rev_growth_yoy, coupon_used_for_payment, Days_since_connect and cashback are fetched as objects. It represents the presence of non-numerical values in the numerical fields. So, the data has been further drilled down by filtering it as a unique value in Understanding the Data reference code (DU4).

To identify the unique values in the dataset (DU4)

```
for col, ro in df.items():          # iterate each columns and rows through
  print("Feature Name :", col, "        Data Type :", df[col].dtype, "\n")
  print(df[col].unique(), "\n")       # print unique values
  print("_____________________________________________________________________")
```

```
['Debit Card' 'UPI' 'Credit Card' 'Cash on Delivery' 'E wallet' nan]

_____________________________________________________________________
Feature Name : Gender          Data Type : object

['Female' 'Male' 'F' nan 'M']

_____________________________________________________________________
Feature Name : Service_Score          Data Type : float64

[ 3.  2.  1. nan  0.  4.  5.]

_____________________________________________________________________
Feature Name : Account_user_count          Data Type : object

[3 4 nan 5 2 '@' 1 6]

_____________________________________________________________________
Feature Name : account_segment          Data Type : object

['Super' 'Regular Plus' 'Regular' 'HNI' 'Regular +' nan 'Super Plus'
 'Super +']
```

Figure 3: Invalid character identification

The above Figure depicts the presence of invalid characters and the need to fix the data inconsistencies.

Missing Value Detection

As there were some null values found in the section in Understanding the Data reference code (DU2), a question of feature consideration popped up. That resulted in the identification of the percentage of null values.

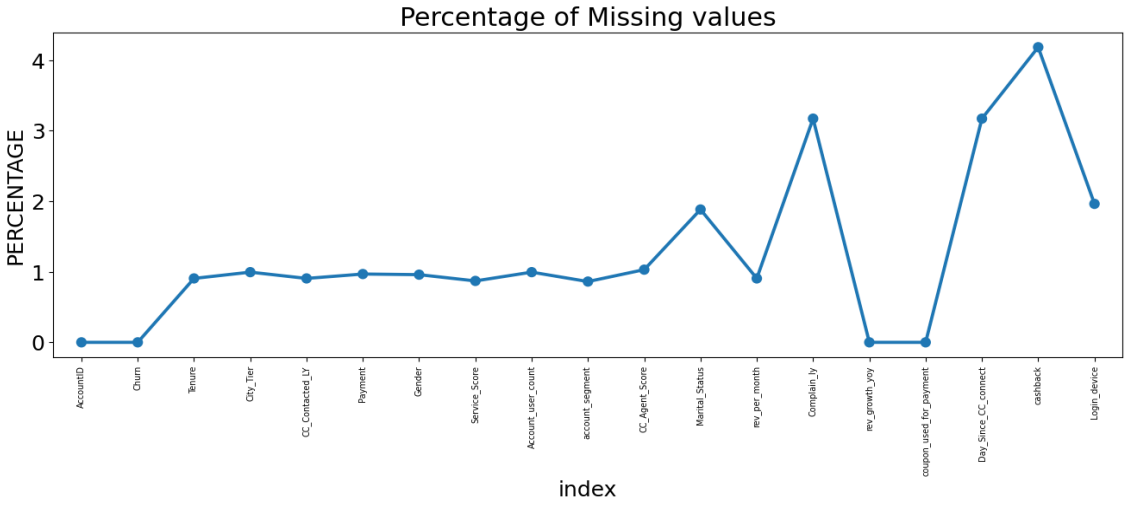

Figure 4:Percentage of missing data

To identify the percentage of missing values the code in the section Understanding the Data (DU6) has been used. The code calculates the percentage of missing values and plots it in a line graph with points (point plot). Even though there is no standard percentage to consider a feature in the analyses. But a percentage of 5 to 10 is acceptable. In our case, we can consider all the features as it has less than 6 per cent of null values.

Feature Removal

AccountID is a unique identification number given for each record. It doesn't add value, so the AccountId field is removed. After removing the AccountID, we have 18 features. The Shape of the data for analysis became (11260,18) – Coded under Feature Engineering reference code (FE1).

Data Cleaning

While analyzing the data, there was a presence of invalid characters and inconsistencies in the dataset. The raw data contained invalid symbols like **'*','&', '$', '+', '@', and '#'** in various columns. They have been replaced with Null values. The **'&&&&'** symbol in the Login_device column has been replaced with **"Others".** The program under the section Feature Engineering reference code "Data Cleaning (FE2)" removed all the invalid characters. After removing the invalid characters. we have a clean dataset of numerical fields and categorical fields.

Feature Name Correction

For clarity in the Feature names, a few labels are renamed for better understanding as below. The same can be seen in the Feature Engineering reference code FE3 section of the Feature Engineering.

```
Index(['Churn', 'Tenure', 'City_Tier', 'Contacted_CC_in_lst_12M', 'Payment',
       'Gender', 'Service_Rating', 'Account_user_count', 'account_segment',
       'Customer_CC_Rating', 'Marital_Status', 'Avg_Revenue_per_Mnth',
       'Complaint_recd_L_Yr', 'Percent_Annual_rev_growth',
       'coupon_used_for_payment', 'Day_Since_CC_connect', 'cashback',
       'Login_device'],
```

Figure 5: Feature name correction

After renaming the features, they are categorized into two types of fields. The program snipper under the section in Feature Engineering reference code "Data Cleaning (FE2)" will create lists of categorical (cat_fld) and numerical fields (num_fld) for target operations.

Missing Value Treatment

The Null values are treated with basic statistical measures. All the null values in numerical fields are replaced with mean and categorical fields are replaced with mode. The output can be found in the coding section in the Feature Engineering reference code “Missing value treatment (FE4)”.

```
Missing value counts after missing value treatment
Categorical fields
__________________
Payment                0
Gender                 0
account_segment        0
Marital_Status         0
Login_device           0
City_Tier              0
Service_Rating         0
Account_user_count     0
Customer_CC_Rating     0
Complaint_recd_L_Yr    0
Churn                  0
dtype: int64

Numerical fields
__________________
Tenure                          0
Contacted_CC_in_lst_12M         0
Account_user_count              0
Avg_Revenue_per_Mnth            0
Percent_Annual_rev_growth       0
coupon_used_for_payment         0
Day_Since_CC_connect            0
cashback                        0
dtype: int64
```

Figure 6: Missing value counts after treatment

Outlier Detection & Removal

Outliers are the extreme values present in the numerical fields.

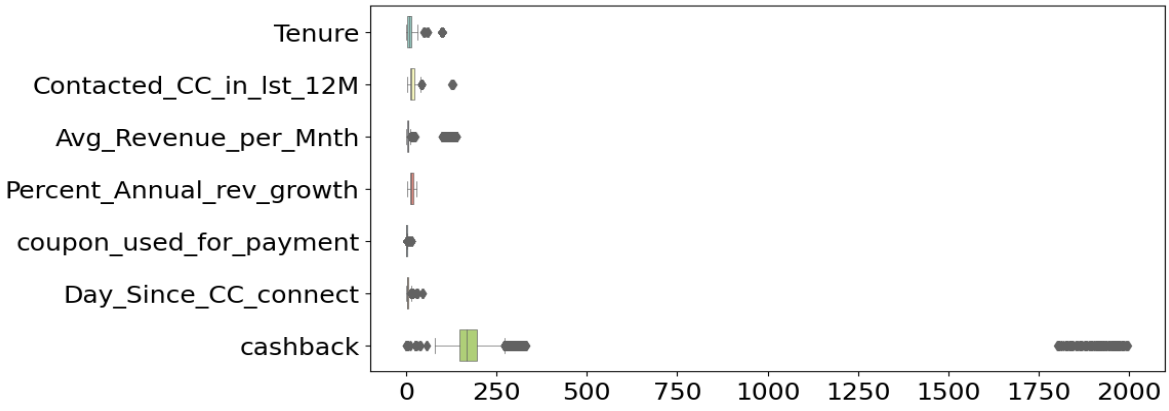

Figure 7: Plot showing the Outliers before treating

The above Plot depicts the presence of Outliers in the cleaned data. Among all the attributes, the CC_Contacted_Ly, rev_per_month, and cashback fields contain outliers. Outliers concerning the Cashback field contain the majority of the outliers and are the most extreme.

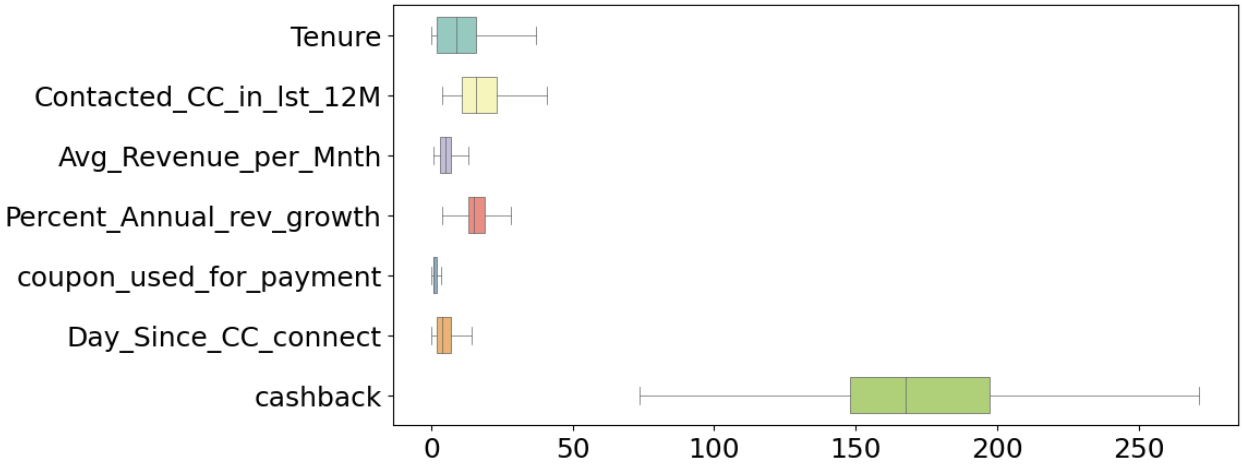

Figure 8: Plot showing the Outliers after treating

In the above plot, all the outliers that are above the Maximum and Minimum values of the box plot are replaced by the Maximum and Minimum values respectively for each box plot.

	count	mean	std	min	25%	50%	75%	max
Churn	11260.0	0.168384	0.374223	0.00	0.00	0.00	0.00	1.00
Tenure	11260.0	10.290142	8.887725	0.00	2.00	9.00	16.00	37.00
City_Tier	11260.0	1.647425	0.912763	1.00	1.00	1.00	3.00	3.00
Contacted_CC_in_lst_12M	11260.0	17.833126	8.562396	4.00	11.00	16.00	23.00	41.00
Service_Rating	11260.0	2.903375	0.722476	0.00	2.00	3.00	3.00	5.00
Account_user_count	11260.0	3.710790	0.924278	1.50	3.00	4.00	4.00	5.50
Customer_CC_Rating	11260.0	3.065808	1.372663	1.00	2.00	3.00	4.00	5.00
Avg_Revenue_per_Mnth	11260.0	5.321048	2.884834	1.00	3.00	5.00	7.00	13.00
Complaint_recd_L_Yr	11260.0	0.276288	0.447181	0.00	0.00	0.00	1.00	1.00
ercent_Annual_rev_growth	11260.0	16.193339	3.757222	4.00	13.00	15.00	19.00	28.00
coupon_used_for_payment	11260.0	1.475577	1.102254	0.00	1.00	1.00	2.00	3.50
Day_Since_CC_connect	11260.0	4.609858	3.482954	0.00	2.00	4.00	7.00	14.50
cashback	11260.0	178.575979	43.653108	73.76	147.89	167.49	197.31	271.44

Figure 9: Basic statistics after data cleaning

Now the data is cleaned, removing inconsistencies and outliers. To have a look at the distribution run the “df.describe.T” code under the section Feature Engineering reference code (FE6). The above figure shows the output. The distribution seems to be clean. The data is now ready to go ahead with Exploratory Data Analysis (EDA).

EXPLORATORY DATA ANALYSIS

EDA is one of the foremost in any data analysis to understand if there are any outliers present in the data and how different variables are related to each other and helps in designing statistical analysis that produces meaningful results.

It is a method used to investigate, analyze and summarize data sets and their main characteristics, often employing data visualization methods. It helps in making it easier for data scientists to discover patterns, spot anomalies, test a hypothesis, or check assumptions.

It is primarily used to see what data can reveal beyond the formal modelling or hypothesis testing task and provides a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate.

It can help answer questions about standard deviations, categorical variables, and confidence intervals. Once EDA is complete and insights are drawn, its features can then be used for more sophisticated data analysis or modelling, including Machine Learning.

Various statistical functions and techniques that can be performed with EDA tools include but are not limited to

- Clustering and dimension reduction techniques, help create graphical displays of high-dimensional data containing many variables.
- Univariate visualization of each field in the raw dataset, with summary statistics.

- Bivariate visualizations and summary statistics allow you to assess the relationship between each variable in the dataset and the target variable you're looking at.
- Multivariate visualizations, for mapping and understanding interactions between different fields in the data.
- K-means Clustering is a clustering method in unsupervised learning where data points are assigned into K groups, i.e., the number of clusters
- Predictive models, such as linear regression, use statistics and data to predict outcomes.

EDA is also used to:

1, Identify variable distribution

2. Plot various graphs like Histograms, box plots, scatter plots, etc.

3. Analysis of the Correlation of different variables in the problem statement.

Univariate Analysis (EDA1)

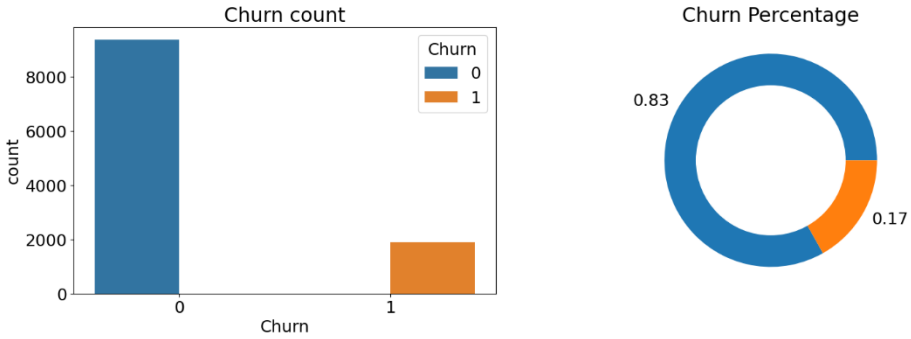

Figure 10: Plot describing the Customer churn percentage

Among all the users in the company (11260) about 17% of the users churned.

Numerical Field Analysis (EDA2)

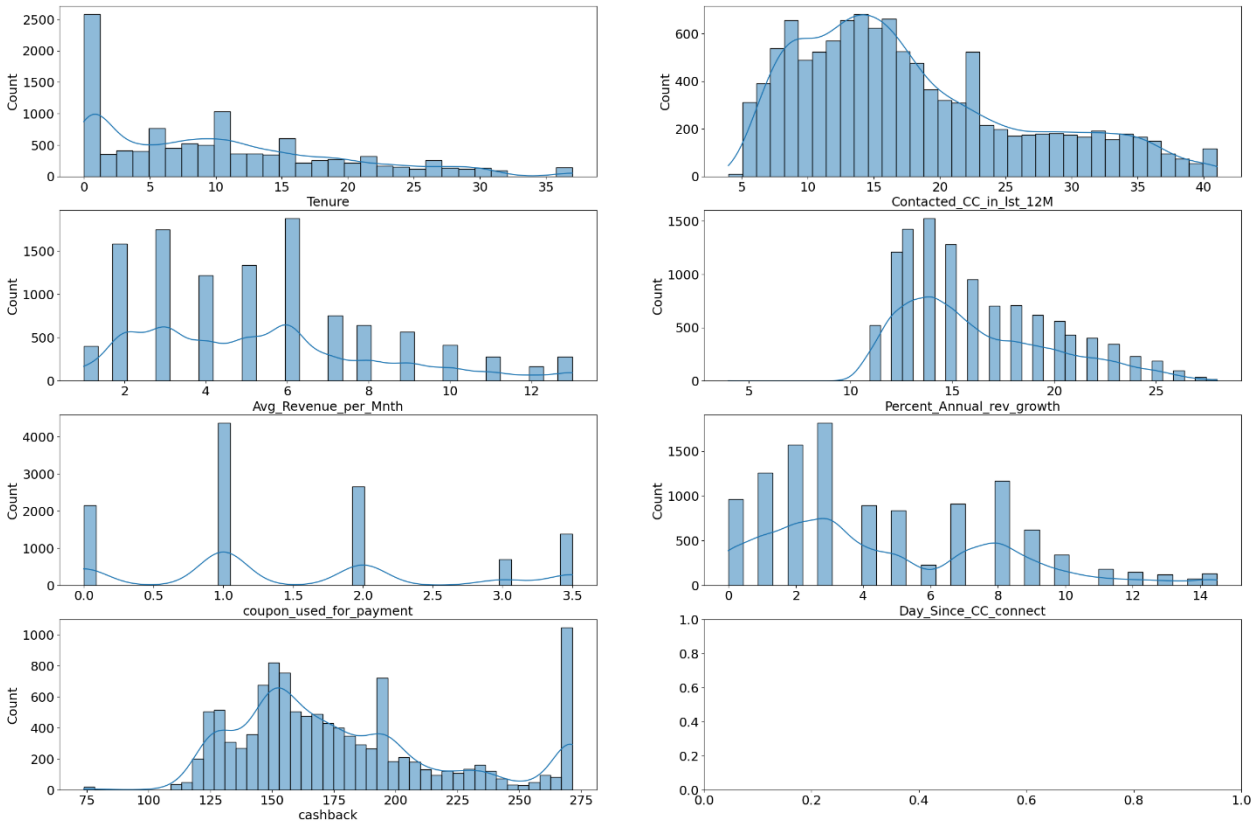

Figure 11: Plot showing the distribution of numerical attributes

From the above bar chart, it is observed that:

- The majority of Customers have tenure in the range of 1-20 Months.
- The majority of customers contacted Customer Care about 5-25 times in the last 12 months
- The average revenue growth per customer is about 15
- The average Cashback obtained by customers is about 150

Categorical Field Analysis (EDA3)

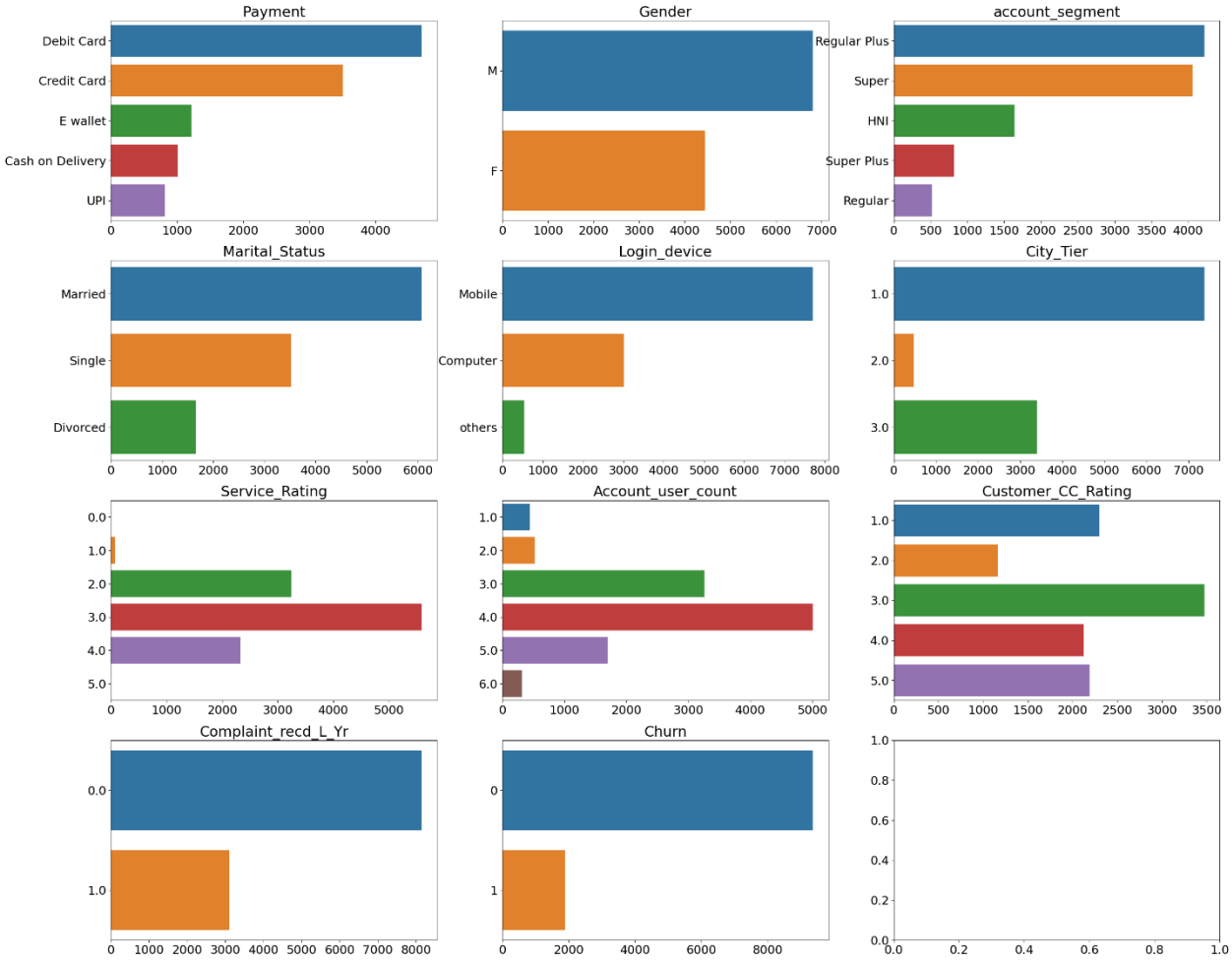

Figure 12: Plot showing the categorical representation of the data after cleaning

From the above bar charts, it is observed that:

- The majority of users are choosing a Debit Card and Credit Card as their payment mode.
- Significantly more no. of male users are there when compared to female users.
- Among all the Account segments available most of the Users belong to the “Regular Plus” and “Super”
- About 50% of the users are married.
- A large portion of the users are using “Mobile” as their login device
- About 99% of the users belong to Tier 1 and Tier 3 Cities
- The Majority of Users rated the service score as 3/5 which signifies “Average”.
- The majority of user accounts are being used by 4 members.

Categorical Field Analysis (EDA5)

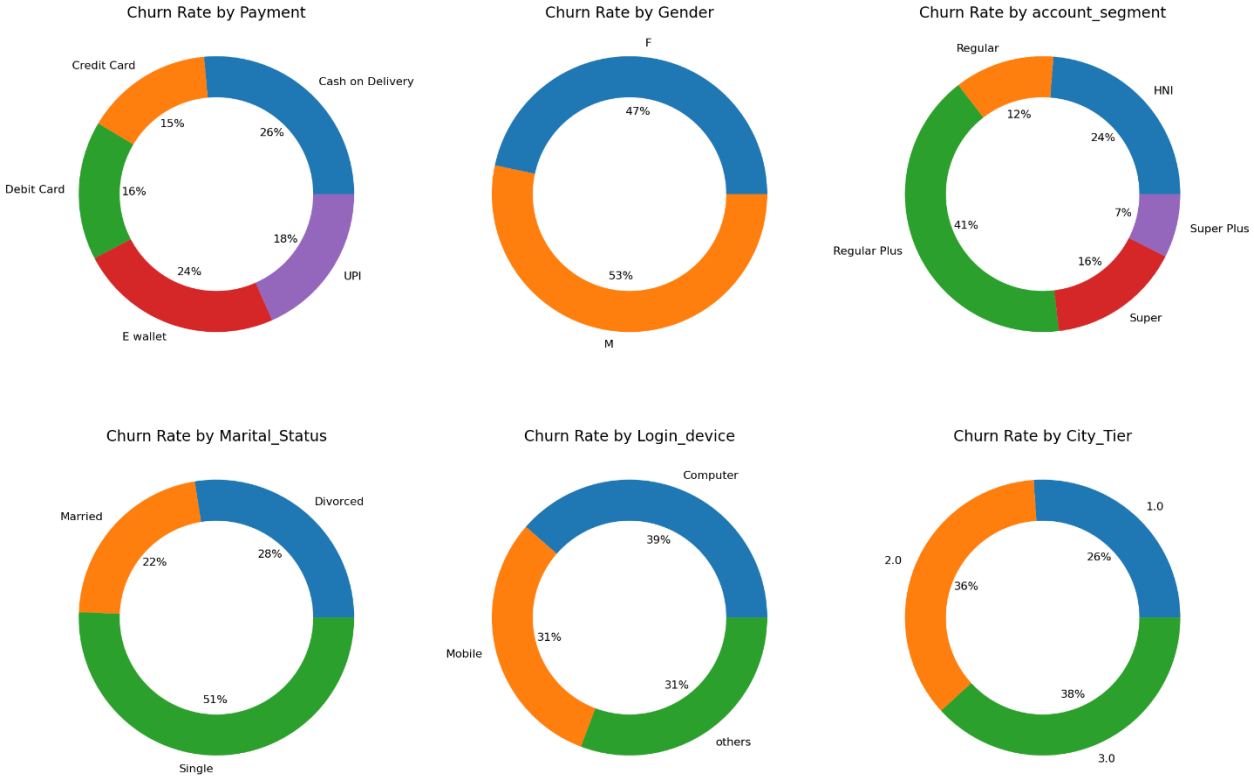

Figure 13: Results of Univariate Analysis of Churned Users

The above plot represents a few observations about the Churned users:

- Even though all the payment modes are used by users, the maximum number of users used Cash on delivery as their payment mode (26%)
- The majority belong to account segments “Regular Plus” (41%) and “HNI” (24%)
- About 51% have their Marital Status as “Single”
- The login devices used by churned users are Computer (56%) and Mobile (44%)
- Tier 1 City (26%), Tier 2 City (36%), Tier 3 City (38%)

Bivariate Analysis

Numerical Field Analysis (EDA4)

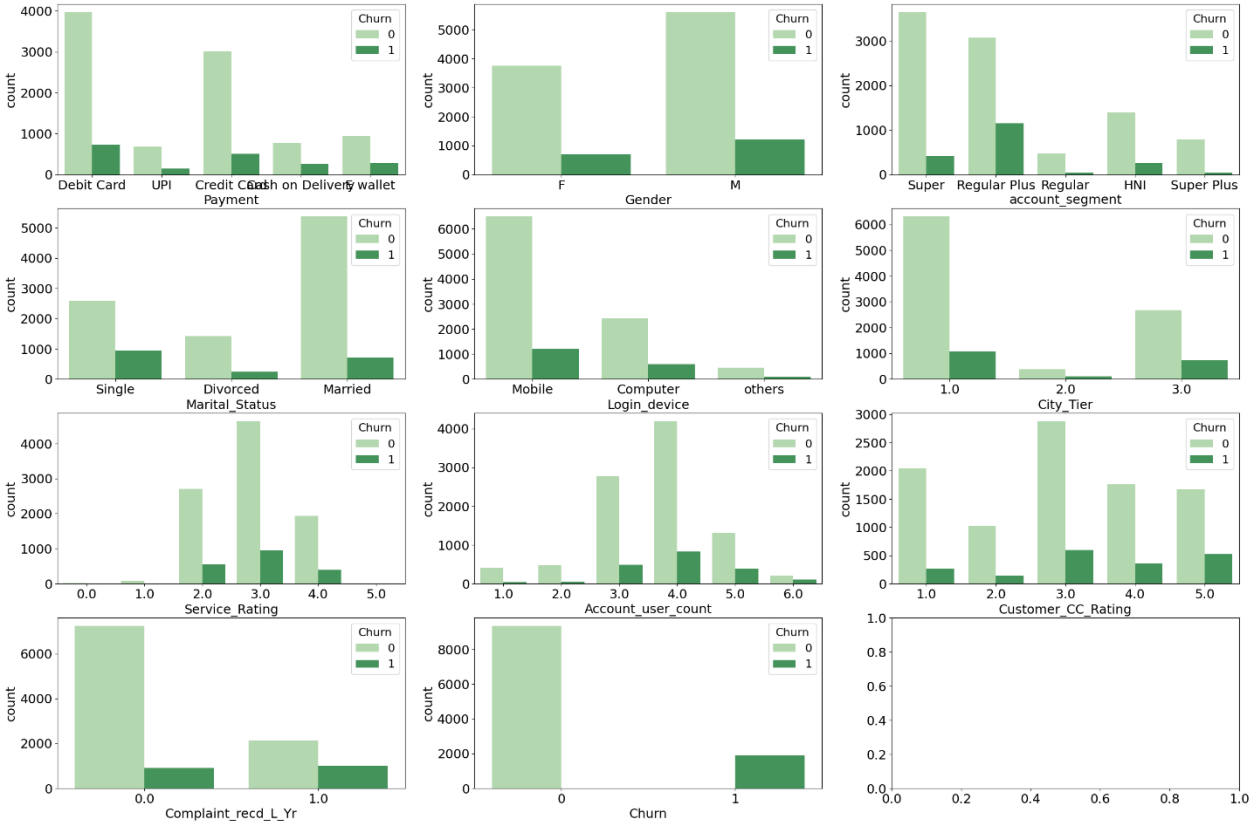

Figure 14: Plot that gives insights into Bivariate Analysis of numerical fields of churned users

The above plot shows the details of the Churned users:

- The majority of the Churned users belong to the account segment of “Regular Plus” and “Super”
- The majority of Churned users used Mobile as their login device
- The majority of Churned users have given a service score of 3/5
- The majority of Churned user accounts have 4 users per account

Multivariate Analysis (EDA6)

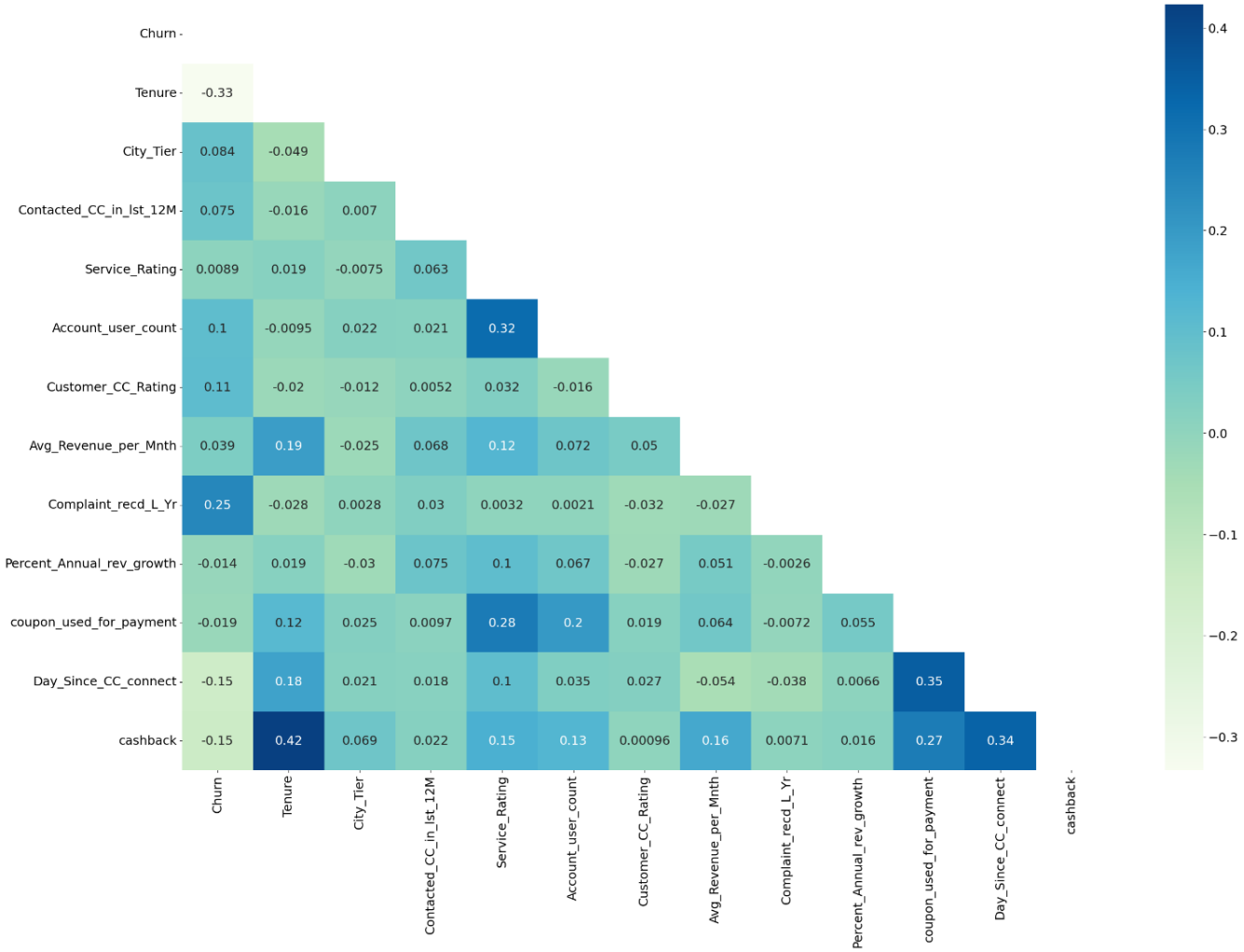

Figure 15: Correlation Plot of various attributes

The above correlation plot established a Positive Correlation between:

1. Churn – Complain_ly (0.25)
2. Tenure – Cashback (0.42)
3. Service_Score – Account_User_Count (0.32)
4. Coupon_Used_for_payment – Day_Since_CC_connect (0.35)
5. Day_Since_CC_Connect – (0.34)

And a Negative Correlation between:

1. Churn – Tenure (-0.33)
2. Churn – Day_Since_CC_Connect (-0.15)
3. Churn – Cashback (-0.15)

MODEL BUILDING AND INTERPRETATION

The thorough Exploratory data analysis derived valuable insights that show the data is cleaned and ready to undergo machine learning application. As a standard machine learning approach data preprocessing should be conducted before applying any machine learning model. Linear Discriminant Analysis (LDA), Logistic Regression, K-Nearest Neighbor classifier (KNN), Support Vector Machine, Random Forest, and Extreme Gradient Boosting (XGBoost) have been taken for the study. Both are supervised learning classification algorithms. As a standard procedure, before applying machine learning data preprocessing has been done to encode the categorical fields to numeric and standardize the data for better sailing.

Data Preprocessing

The dataset contains 11,260 records and 18 features. It is under the programming section “Data Pre-Processing (ML2)”. Then the label encoding has been done to convert text features to numeric ones in the coding section “Label encoding (ML2A)”. The target variable “Churn" has been separated from the dataset to a new DataFrame called (Y), and the remaining features as called in (X) DataFrame and these are the predictors.

The Standardization (z-score) has been applied to the features to ensure uniform scaling in section Feature Standardization (ML3). Finally, the data has been split into a training set (70%) and a test set (30%) in the coding section “Train Test split 70:30 (ML4)”. The training set contains 7,882 records, and the test set contains 3378 records. The next step is to train the models using the train dataset and predict. So, the following models are chosen for the study.

MACHINE LEARNING

Machine learning is and branch of Artificial Intelligence leveraging historical data to make predictions. Machine learning uses programmed algorithms to analyze input data to predict output values. There are different types of algorithms used in machine learning such as supervised,

unsupervised and reinforced. A supervised learning algorithm uses labeled data whereas the unsupervised learning algorithm uses unlabeled to learn and predict the output. The reinforced learning uses trial and error learning methodology to interact with the environment and predicts based on maximum rewards. Rewards are received for positive actions and penalties for harmful actions. Classification and regression fall under the umbrella of supervised learning. Customer churn prediction is a classification problem using supervised learning to predict the outcome. To solve the customer churn problem, some of the chosen Machine Learning models are Random Forest, Support Vector Machine (SVM), XGB Classifier, KNN, Logistic Regression and Linear Discriminant Analysis (LDA)

- **Linear Discriminant Analysis (LDA)**

  Linear Discriminant Analysis (LDA) is a supervised statistical technique used for dimensionality reduction and classification. It aims to reduce the dimensionality of data while maintaining class separability. LDA requires labelled data and is ideal for scenarios where you need to distinguish between different classes. The process involves calculating class mean vectors, computing within-class and between-class scatter matrices, and solving an eigenvalue problem to obtain discriminant vectors. These vectors define a lower-dimensional feature space into which the data is projected.

  LDA is particularly useful in multi-class problems where it reduces the feature space while preserving class distinctions. Common applications include face recognition, text classification, and biomedical data analysis. LDA is supervised and emphasizes class separability rather than maximizing variance. It serves as a powerful tool for enhancing data analysis and classification when you have labelled data.

- **Random Forest**: Random Forest is a popular supervised machine learning algorithm used for both classification and regression. It involves the concept of ensemble learning which combines multiple weak classifiers to resolve a complex problem. It uses numerous Decision Tree classifiers (weak classifiers) and averages the outcome to predict the target. It prevents the problem of overfitting by increasing the number of trees.

- **Support Vector Machine:** Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression problems. It classifies the data by identifying a hyperplane between two classes, so the hyperplane is called the best decision boundary between classes. To create a hyperplane SVM chooses the extreme points/vectors which are the data points lying close to the hyperplane. These extreme cases are called as support vectors, and hence algorithm is known as the Support Vector Machine.

- **Extreme Gradient Boosting (XGBoost)**: Extreme gradient boosting (XGBoost) is a popular machine learning model that uses ensemble learning models in predicting. Like Random Forest it uses decision trees and boosting. Boosting is an ensemble learning technique to increase the accuracy of the model. XGBoost builds trees sequentially and each updates the residual errors by learning from its predecessors called boosting. It also uses a gradient descent function to minimize the loss function in each build.

The moto of using the various models is to find the best - performing model.

Training the models for prediction

To train the chosen models, we used the code under the section "Training the models (ML7). The code uses two lists 'models' and 'model_names'. The 'models' holds the list of all the classifiers and the model_name has the name of each model. The loop picks each item from the list and fits the trains the training dataset (x_train) to the model. Once the training is done, it is used to predict y target. The predicted target is used to compare and evaluate the performance of the model.

The code uses a function called "get_performance_score()" to compute accuracy, precision, recall, specificity and f1 score and returns output when it is called in the loop. These are the different metrics used to evaluate the performance of the classification model. when the program was executed, get_performance_score() returned the below output.

```
Using model: Linear discriminant analysis
                 mdl_score     accuracy        recall          precision       specificity     f1_score
Trainning Score:  0.875          0.875          0.394           0.743           0.972           0.515
Test Score:       0.881          0.881          0.412           0.776           0.976           0.538
_____________________________________________________________________________________________________
Using model: Logistic Regression
                 mdl_score     accuracy        recall          precision       specificity     f1_score
Trainning Score:  0.882          0.882          0.446           0.754           0.971           0.560
Test Score:       0.888          0.888          0.460           0.787           0.975           0.581
_____________________________________________________________________________________________________
Using model: K-Nearest Neighbors
                 mdl_score     accuracy        recall          precision       specificity     f1_score
Trainning Score:  0.978          0.978          0.895           0.971           0.995           0.931
Test Score:       0.957          0.957          0.807           0.931           0.988           0.865
_____________________________________________________________________________________________________
Using model: XGBClassifier
                 mdl_score     accuracy        recall          precision       specificity     f1_score
Trainning Score:  0.999          0.999          0.997           1.000           1.000           0.998
Test Score:       0.971          0.971          0.868           0.954           0.991           0.909
_____________________________________________________________________________________________________
Using model: Support Vector Machine
                 mdl_score     accuracy        recall          precision       specificity     f1_score
Trainning Score:  0.944          0.944          0.710           0.945           0.992           0.811
Test Score:       0.928          0.928          0.616           0.936           0.991           0.743
_____________________________________________________________________________________________________
Using model: Random Forest
                 mdl_score     accuracy        recall          precision       specificity     f1_score
Trainning Score:  1.000          1.000          1.000           1.000           1.000           1.000
Test Score:       0.973          0.973          0.858           0.978           0.996           0.914
_____________________________________________________________________________________________________
```

Figure 16: Performance score after training model for first time

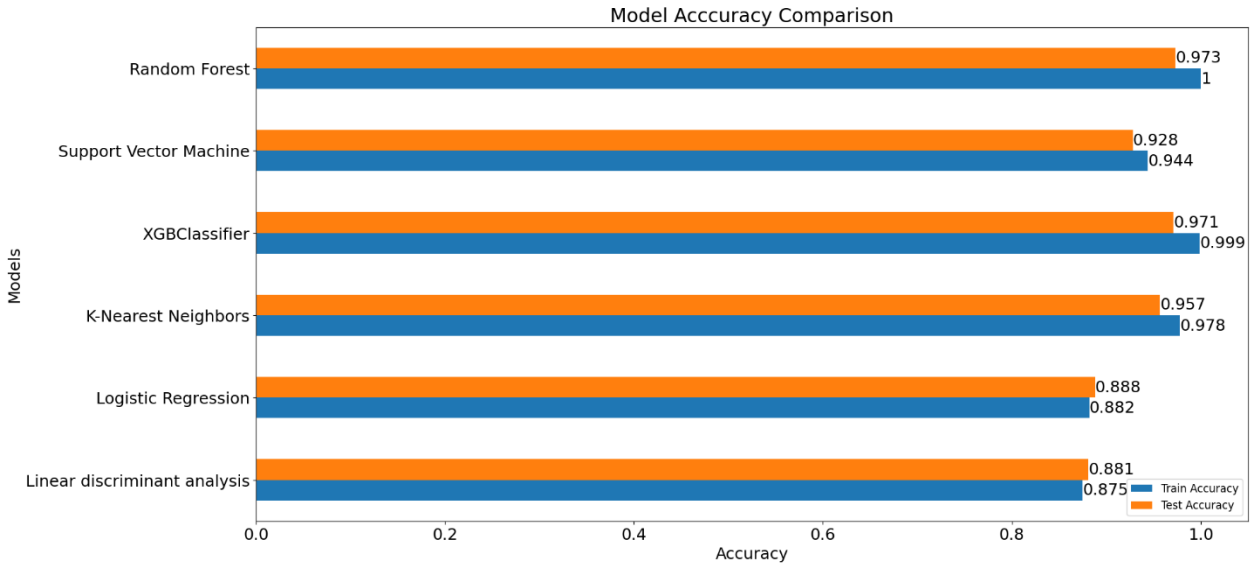

Figure 17: Test, Train accuracy comparison for chosen ML models

The initial run returns the above figure which depicts the performance of the models. Even though most of the evaluation metrics are present, some models are chosen based on the accuracy for

further analyses. Random Forest, XGB Classifier (XGBoost), Support Vector Machine and KNN for AUC and ROC analysis.

Model Evaluation

After the initial analysis, some of the models are chosen for further drill-down. AUC ROC curves have been taken to get a clear picture of model performance. some of the metrics used in model evaluation are accuracy, precision, recall, f1-Score and AUC score. To calculate the metrics. We need to get True Positive (TP), True Negative (TN), False Negative (FN), and False Positive (FP)

- when a classification model correctly predicts a positive outcome, it is called TP.
- when a classification model incorrectly predicts a negative outcome, it is called FN.
- when a classification model correctly predicts a negative outcome, it is called TN.
- when a classification model incorrectly predicts a positive outcome, it is called FP.
- Accuracy is the fraction of correct prediction made by the model. It is calculated using the formula.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Recall**, also known as sensitivity, is the fraction of actual positive cases that are correctly identified as positive by the model.

$$Recall = \frac{TP}{TP + FN}$$

- Precision is the fraction of correct positive predictions.

$$Precision = \frac{TP}{TP + FP}$$

- **F1-score** is the harmonic mean of precision and recall. It is a balanced measure of the model's ability to identify both positive and negative cases correctly.

- The AUC ROC curve is calculated by plotting the True Positive Rate against the False Positive Rate. An AUC of 1.0 indicates that the model is perfect, and an AUC of 0.5 indicates that the model is no better than random guessing. In general, an AUC of 0.7 or higher is considered to be good performance.

K-Nearest Neighbors

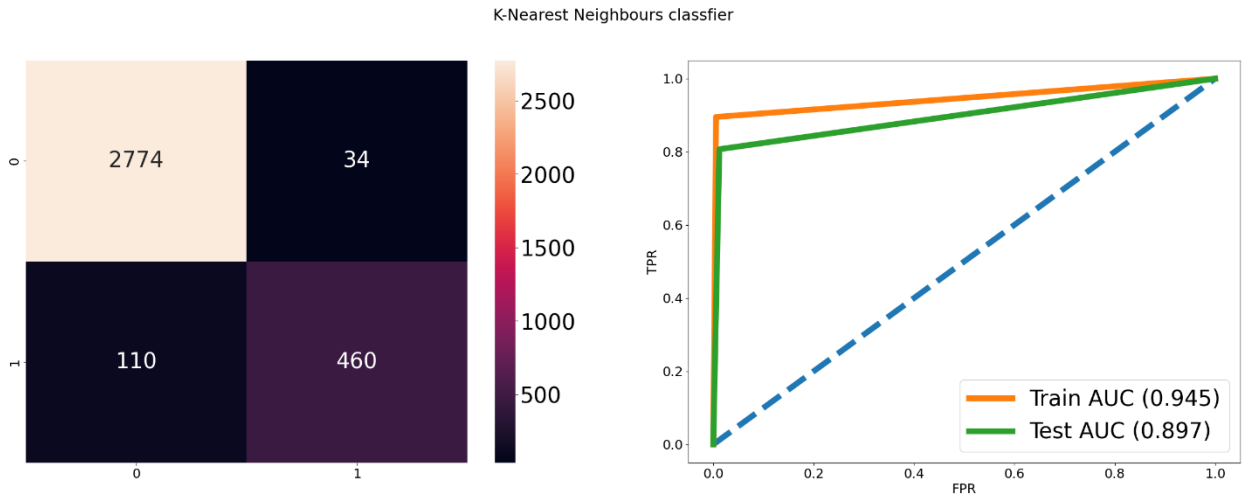

Figure 18 : Confusion Matrix and AUC curve for KNN

Table 2: Performance score for KNN classifier

Scores	accuracy	recall	precision	specificity	f1_score	AUC_score
Test	0.957	0.807	0.931	0.988	0.865	0.897455
Train	0.978	0.895	0.971	0.995	0.931	0.94484

The performance of the K-Nearest Neighbors classifier on the test set is good, with an AUC score of 0.897. This means that the model can distinguish between positive and negative cases relatively well. There is some evidence of overfitting in the model, as the training accuracy is higher than the test accuracy (0.978 vs. 0.957). This suggests that the model is learning the specific patterns in the training data too well, and is not able to generalize to new data as well. Overall, the model is performing well on the test set, but there is some evidence of overfitting. It may be helpful to try to reduce the overfitting by using a more complex model, or by using regularization techniques.

Extreme Gradient Boosting

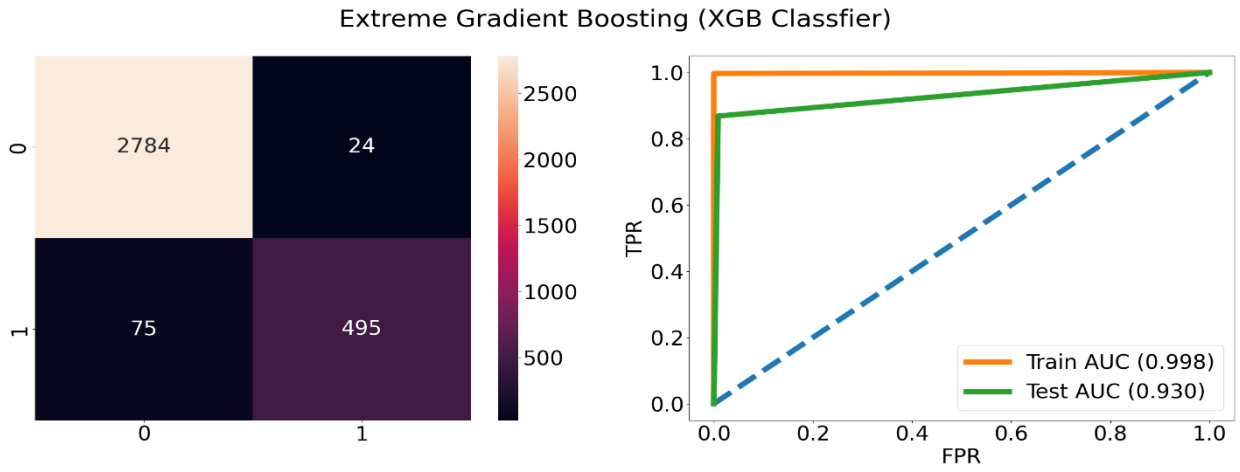

Figure 19: Confusion Matrix and AUC curve for Extreme Gradient Bosting (XGBoost)

Table 3: Performance score for Extreme Gradient Boosting (XGBoost)

Scores	accuracy	recall	precision	specificity	f1_score	AUC_score
Test	0.971	0.868	0.954	0.991	0.909	0.929937
Train	0.999	0.997	1	1	0.998	0.99849

Based on the result, the model is likely overfitting the training data. The training accuracy is very high (0.999), but the test accuracy is significantly lower (0.971). This suggests that the model is learning the specific patterns in the training data too well and is not able to generalize to new data as well. Another indicator of overfitting is the difference between the training and test AUC scores. The training AUC score is 0.998, while the test AUC score is 0.929. This difference is larger than 0.05, which is generally considered to be a sign of overfitting.

To reduce overfitting, you could try the following:

- Use a simpler model.
- Use a smaller training set.
- Use regularization techniques, such as L1 or L2 regularization.
- Use data augmentation techniques to create more training data.

Overall, the model is performing well on the test set, but there is some evidence of overfitting. By taking steps to reduce overfitting, you may be able to improve the overall performance of the model

Support Vector Machine

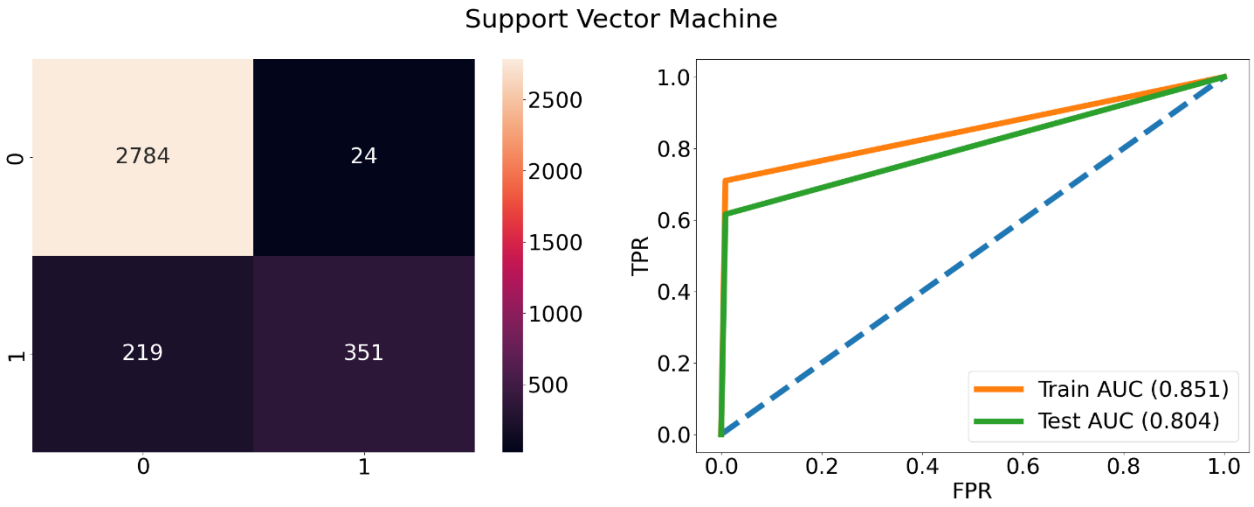

Figure 20: Confusion Matrix and AUC curve for Support Vector Machine (SVM)

Table 4: Performance score for Support Vector Machine (SVM)

Scores	accuracy	recall	precision	Specificity	f1_score	AUC_score
Test	0.928	0.616	0.936	0.991	0.743	0.803621
Train	0.944	0.71	0.945	0.992	0.811	0.850632

Based on the result, the model is likely underfitting the training data. The training accuracy is relatively low (0.944), and the test accuracy is even lower (0.928). This suggests that the model is not able to learn the patterns in the training data well enough and is therefore not able to generalize to new data. Another indicator of underfitting is the difference between the training and test AUC scores. The training AUC score is 0.85, while the test AUC score is 0.803. This difference is less than 0.05.

Random Forest

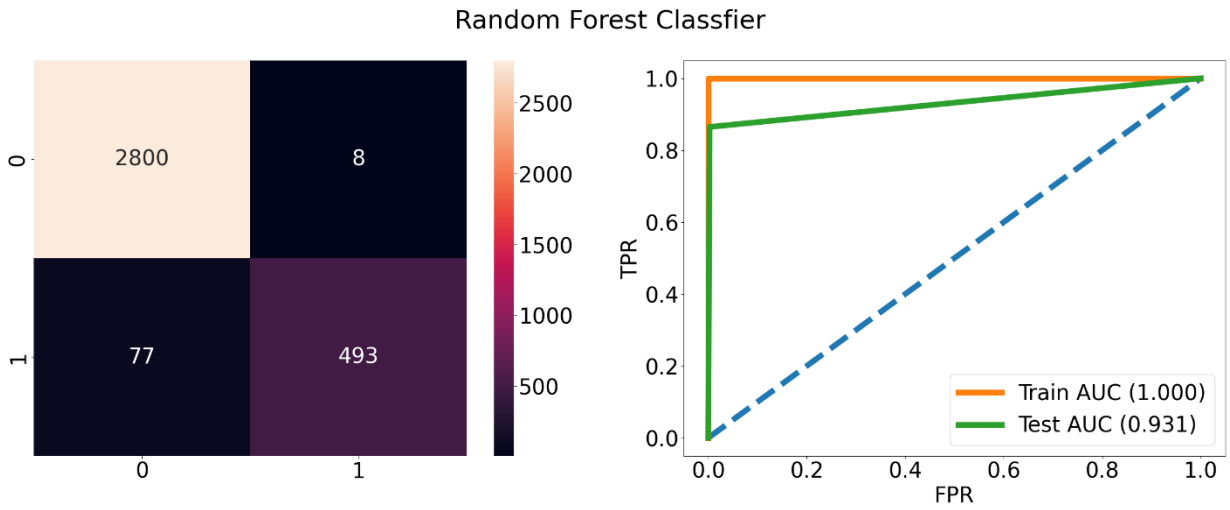

Figure 21: Confusion Matrix and AUC curve for Random Forest

Table 5: Performance Score for Random Forest

Scores	accuracy	recall	precision	specificity	f1_score	AUC_score
Test	0.975	0.865	0.984	0.997	0.921	0.931032
Train	1	1	1	1	1	1

Based on the image, the model is likely overfitting the training data. The training accuracy is very high (1.00), but the test accuracy is significantly lower (0.97). This suggests that the model is learning the specific patterns in the training data too well and is not able to generalize to new data as well.

Another indicator of overfitting is the difference between the training and test AUC scores. The training AUC score is 0.99, while the test AUC score is 0.93. This difference is larger than 0.05, which is generally considered to be a sign of overfitting.

To reduce overfitting, you could try the following:

- Use a simpler model.
- Use a smaller training set.

- Use regularization techniques, such as L1 or L2 regularization.
- Use data augmentation techniques to create more training data.

Overall, the model is performing well on the test set, but there is some evidence of overfitting. By taking steps to reduce overfitting, you may be able to improve the overall performance of the model.

Model Optimization

Model optimization is done to improve the efficiency and performance of the machine learning models. Hyper Parameter Tuning is the technique used for model optimization. A cross validation is done to ensure the consistency of the prediction, it splits the entire data set to k number of folds and run model training and prediction and the average is calculated for k number of folds. Where k is number of folds we can pass while declaring the object. If the value is not passed then it will take 5 as a default value.

Cross Validation: From the analysis, it is observed that a hyper parameter turning can optimize the performance of the models. The codes executed under the section "Cross validation (MLCV)" run the cross-validation of the best performing model XGBoost resulting in a consistent average score across all the five folds of the data fed.

```
Cross-Validation Scores: [0.96536412 0.97646536 0.97424512 0.97424512 0.97380107]
Mean Accuracy: 0.9728241563055061
Standard Deviation: 0.003844557872453515
AUC Scores: [0.99196328 0.99338045 0.99426653 0.98879508 0.99192947]
Mean AUC: 0.9920669643244633
Standard Deviation of AUC: 0.0018600104737306292
```

Figure 22: Cross Validation Score for XGBoost

Figure 22 shows the AUC score of 99.2% shows the best performance in predicting the test scores.

Hyper Parameter Tuning

As the scores are consistent it was decided to do the hyperparameter tuning under the coding section "Hyper Parameter Tuning (MLHPT)". However, it is observed from literature surveys that most of the study reveals a tuning the parameters rewards the best results. So, it was decided to run all the models through hyperparameter tunning. The GridSearchCV was used in the codes to perform the hyperparameter tuning. As the codes were executed, it split the entire data into different subsets of folds based on the parameter fed to the GridSearchCV model. The ability of GridSearchCV to do the cross-validation using the parameter passed has been utilized fully in quest of the best estimator. A combination of the different parameters was declared for each model under the coding section "Declaring parameters for the models (MLHPT1)"

For **KNN**, we requested the GridSearchCV to use both metrics Euclidean and Manhattan distance and also asked to check the result in different k values such as 3,5,7,9.

For **Random Forest**, a random number of estimators are chosen such as 10, 101 and 200. Also asked the gird to search different criteria 'gene' and 'entropy, the former uses feature bagging which reduces the correlation between trees in the forest. The latter deals with the impurity or randomness of nodes in the decision trees.

For the **Support Vector Machine**, initially, we used we initially used kernel poly, 'rbf' and sigmoid. Out of the 3 kernels 'rbf' returned best estimator when tested individually. As the code was running slow when tested, we later decided to use the best performing 'rbf' so we commented all other kernels.

For **XGBoost,** we have passed the parameter as several estimators as 100, 200 and 300. Also, the different learning rates are 0.1, 0.01, 0.02 and 0.03. And asked the grid to check the result at different max_depth.

When the code was executed with the above parameters, it split the dataset into (5 * k) parameters where 5 is the number of folds and k is the number of different parameters.

```
Tuning KNN…
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Tuning RF…
Fitting 5 folds for each of 18 candidates, totalling 90 fits
Tuning SVM…
Fitting 5 folds for each of 9 candidates, totalling 45 fits
Tuning XGB…
Fitting 5 folds for each of 48 candidates, totalling 240 fits
```

Figure 23: Hyper Parameter Tuning folds, number of parameters and total fits

As the XGBoost was performing well in the previous tests, the number of parameters was increased to get the best estimator and a total of 240 fits with different parameters. While the loop was running on each model, cross-validation was done with different parameters, And the result was collected at each iteration such as best estimator, best score and the test scores such as accuracy, f1 score and AUC ROC scores in a dataset.

A report was generated for the performance evaluation section "Getting best estimators and scores from the grid (MLHPT3)" of the code. And it returned the below output.

```
KNN:
Best parameters: {'metric': 'manhattan', 'n_neighbors': 7, 'weights': 'distance'}
Best cross-validation score: 0.9876
Test scores:
accuracy: 0.9772
f1: 0.9297
roc_auc: 0.9436
_____________________________________________________________________________________
RF:
Best parameters: {'criterion': 'entropy', 'max_depth': 6, 'n_estimators': 101}
Best cross-validation score: 0.9264
Test scores:
accuracy: 0.9065
f1: 0.6580
roc_auc: 0.7578
_____________________________________________________________________________________
SVM:
Best parameters: {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}
Best cross-validation score: 0.9842
Test scores:
accuracy: 0.9769
f1: 0.9301
roc_auc: 0.9505
_____________________________________________________________________________________
XGB:
Best parameters: {'learning_rate': 0.2, 'max_depth': 9, 'n_estimators': 300}
Best cross-validation score: 0.9873
Test scores:
accuracy: 0.9790
f1: 0.9348
roc_auc: 0.9447
_____________________________________________________________________________________
```

Figure 24: Report after Hyper Parameter tuning, Best Parameter, Cross Validation score, accuracy, f1 score, AUC ROC score

From the result, it is evident that the XGBoost classifier outperforms other models with a best estimator "{'learning_rate': 0.2, 'max_depth': 9, 'n_estimators': 300}", accuracy 98%, f1 score 93% and AUC score 94%.

CHAPTER 4

FINDINGS, RECOMMENDATIONS AND CONCLUSION

Findings Based on Observations

Overall, the XGBoost algorithm performed the best on the test set, with an accuracy of 97.90%, f1 score of 93.48%, and ROC AUC of 94.47%.

It is also worth noting that the KNN and SVM algorithms also performed very well, with accuracies above 97.70%. The RF algorithm performed slightly worse than the other three algorithms but still achieved a respectable accuracy of 90.65%.

Overall, the findings suggest that all four of these algorithms can achieve high accuracy on this dataset. However, the XGBoost algorithm appears to be the best choice for this task.

The beauty of hyperparameter tuning was unveiled while performing the tuning operations on the Support Vector Machine.

Table 6:Pre-tuning and Post-Tuning scores for support vector machine

Scores	**accuracy**	**f1_score**	**AUC_score**
Pre-Turning Test Scores	0.928	0.743	0.803621
Post-Turning Test Scores	0.9769	0.9301	0.9505

After the hyperparameter tuning, there was a substantial improvement in overall scoring. This reveals that the tuning process can effectively improve the performance of the machine learning model.

Findings Based on Analysis of Data

Table 7: Performance scores for the models after Hyper Parameter Tuning for analysis

Model_name	**Best_Score**	**Accuracy**	**F1_score**	**roc_auc**
K-Nearest Neighbor	0.9876	0.9772	0.9297	0.9436
Random Forest	0.9264	0.9065	0.658	0.7578
Support Vector Machine	0.9842	0.9769	0.9301	0.9505
Extreme Gradient Boosting	0.9873	0.979	0.9348	0.9447

While comparing the results from hyperparameter tuning and prediction on the test and based on the analysis of the data, the following findings can be made:

- XGBoost has the highest accuracy and F1 score, followed by SVM, KNN and RF.
- SVM has the highest ROC AUC, followed by XGBoost, KNN, and RF.
- Regarding the cross validation score (Best_score) KNN is the highest, followed by XGB, SVM and RF.

Overall, XGBoost is the best-performing model, as it has the second highest ROC AUC and ranks high accuracy and f1 score. KNN is also a good choice, as it has the good accuracy and F1 score. SVM is also a good choice as the overall model scores improved a lot after hyper parameter tuning in terms of accuracy, f1 score and the highest AUC ROC among all the models. RF is the worst-performing model, but it may still be a good choice for certain applications.

General Findings

Key findings from the Given Data and Cleaned Data:

1) The data that is provided has missing values are present in almost all the attributes except AccountID, Churn, rev_growth_yoy, coupon_used_for_payment. And the missing values are in permissible limit (i.e., less than 10%) which recommends we can treat the data and use it as feature for the analysis.
2) Even though there are few outliers present in the data it can be observed that the outliers pertaining to Cashback column is very extreme with a huge gap of above 1300 from the maximum value of the box plot.
3) As the data is representing for about 99 months and the Customer Churn is about 17% we can assume the churning problem is in between initial stage and moderate stage which can be handled if reacted in time and quickly.

4) As the Customers_cc_lst_12M is very high and the average Service Score is about 3/5 we can view the service unit as Average and Service unit has to be Enhanced / Service team shall be subjected to capacity building.
5) Larger portion of the users preferring “Regular Plus” and “Super” as their account segment subscription, large portion of users belongs to “tier 1” and “tier 3” cities and larger portion of Users are having “Mobile” as their login device.
6) About 51% of the churned users are having “Single” as their Marital status which may indicate some requirement to attract and indulge singles in reduce churn rate.
7) We are NOT provided with the Effective start Date (ESD) and Effective End Date (EED) of subscription/AccountID which would facilitate in finding the Customer Churn trend, customer loyalty. i.e., when we have the Tenure, we can only know how many months the customer is active and can’t know during which time period the customer churn is higher.

Recommendation based on findings

Recommendation based on EDA

Upon Diving into the Basic EDA, the following conclusions and recommendations can be drawn:

- We have less than 5% of the users from the Tier 2 cities. It is recommended to run local advertisements and campaigns in Tier 2 cities to spread the wings in Tier 2 cities also.
- About 78% of the users have contacted Customer care more than 10 times in the last 12 months and about 91% of the customers contacted customer care in the last 10 days. It is recommended to introduce a Chatbot/ Interactive Voice Response (IVR) to address and resolve Account info details, and basic problems.
- As the start date of the customer’s account is not provided, by viewing the available data about 63% of the users have had relations with the company for $\leq$ 12 months. So, it is recommended to introduce discounts/offers on the half-yearly and annual packages to improve customer loyalty and customer lifetime.
- It is observed that the user count and churn rate are higher in “Regular Plus” and very low in “Regular” account segments. It is suggested to Club these two account segments

and optimize the pricing accordingly to handle customer churn and attract new customers.

Recommendation based on Machine Learning and Turning:

- As per the machine learning and hyper parameter tuning, Support Vector Machine has shown a substantial improvement in performance scoring. It is advised to test more models that can fine tune a better alternative.

Suggestions for areas of improvement

1. A feature importance can be performed on the model and based on the result, the most contributing feature can be used to re-train the model to enhance the prediction.
2. Other models such as Linear Discriminant Analysis and Logistic Regression were not taken for further analysis due to less accuracy in the initial run, these models also can be used by tuning the parameters to improve the accuracy.

Scope for future research

We have identified the best model which can accurately predict the customer churn. The next level is to segment the customer based on the potentiality of churn as high risk, medium risk and low risk. A recommendation system can be built based on the association and relationship of features. This will enable the organization to tailor the best campaign strategy.

Conclusion

Based on the analysis of the data and further machine learning model evaluation for the customer churn problem, XGBoost is identified as the best-performing model. However, KNN is also a good choice as, as it has the highest accuracy and F1 score. Support Vector Machine is a good choice as well in terms of ROC AUC when compared to other models but, in terms accuracy, f1 score it is less when compared to XGB. Random Forest is the worst-performing model, but it may still be a good choice for certain applications.

References

Hyperparameter Tuning in Python: a Complete Guide. (n.d.). Retrieved from Neptune.ai: https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide

RandomForestClassifier. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Support Vector Machines. (n.d.). Retrieved from www.scikit-learn.org/: https://scikit-learn.org/stable/modules/svm.html

Tharwat, A., Gaber, T., Ibrahim, A., & Hassanien, A. E. (2017). Linear discriminant analysis: A detailed tutorial. *Ai Communications*, 169-190.

XGBoost Documentation. (n.d.). Retrieved from https://xgboost.readthedocs.io/en/stable/

Annexure

Problem Statement - Customer Churn (PS1)

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.

Variables (VD1)

<

Variable	Description
AccountID	account unique identifier
Churn	account churn flag (Target)
Tenure	Tenure of account
City_Tier	Tier of primary customer's city
CC_Contacted_L12m	How many times all the customers of the account has contacted customer care in last 12months
Payment	Preferred Payment mode of the customers in the account
Gender	Gender of the primary customer of the account
Service_Score	Satisfaction score given by customers of the account on service provided by company
Account_user_count	Number of customers tagged with this account

account_segment	Account segmentation on the basis of spend
CC_Agent_Score	Satisfaction score given by customers of the account on customer care service provided by company
Marital_Status	Marital status of the primary customer of the account
rev_per_month	Monthly average revenue generated by account in last 12 months
Complain_l12m	Any complaints has been raised by account in last 12 months
rev_growth_yoy	revenue growth percentage of the account (last 12 months vs last 24 to 13 month)
coupon_used_l12 m	How many times customers have used coupons to do the payment in last 12 months
Day_Since_CC_co nnect	Number of days since no customers in the account has contacted the customer care
cashback_l12m	Monthly average cashback generated by account in last 12 months
Login_device	Preferred login device of the customers in the account

---

Python Libraries Used

---

[259]

0s

```
#INT1  importing Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

---

1. Loading the data

---

[260]

3s

```
# Authorize your drive(select your gmail account and click on the "Allow" button to authorize the colab n
otebook to access your drive)
from google.colab import drive
drive.mount("/content/drive")
```

output

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

---

[261]

4s

```
# Reading the dataset from the drive <use csv or excel on your convenience> uncomment and use it accor
ding to the requirement

# df=pd.read_csv("/content/drive/MyDrive/Customer_Churn_DS/Customer_Churn_Data_new.csv")    # th
ose who use csv file
df=pd.read_excel("/content/drive/My Drive/Capstone_Project_G7/Customer_Churn_Data.xlsx", sheet_na
me='Data for DSBA')

#Those who use jupiter notes upload the spreadsheet and use
#df=pd.read_excel('Customer_Churn_Data.xlsx', sheet_name='Data for DSBA')
```

---

2. Understanding the data

Data Dimension (DU1)

---

[262]

0s

```
df.shape                    #Dimension of the data
```

output

```
(11260, 19)
```

There are 19 features and 11260 records in the dataset.

Understanding the features and feature types (DU2)

[263]

0s

```
print(df.info())              # returns not null value counts, data types of each columns in the dataset
```

output

```
<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

None

---

[264]

0s

```
d={'object':0, 'float':0, 'Integer':0}
for i in df.columns:
  if (df[i].dtype =='int64'):
    d['Integer']+=1
  elif (df[i].dtype =='float64'):
     d['float']+=1
  else:
    d['object']+=1

keys = list(d.keys())
values = list(d.values())
sns.barplot(x=keys, y=values).set(title='Datatype counts')
plt.show()# Show the plot
```

output

As per the basic analysis there are 12 catagorical fields and 11 numerical fields. out of 11 numerical fields there are 2 ineger and 5 decimal columns. While checking the non-null value counts, It is not matching with the total record count. this represents presence of null values in some features.

A glance of the data (DU3)

---

[265]

0s

```
df.head(3)
```

output

While comparing the data against the data type, some numerical filed such Tenure, Account_user_count, Revenue per month, rev_growth_yoy, coupon_used_for_payment, Day since conect and cashback are feteched as object. It

represents the ppresaence of non-numerical values in the numerical fields. Lets furhter drill down the data by filtering it by unique value.

To identify the unique values in the dataset (DU4)

---

[266]

0s

```
for col, ro in df.items():          # iterate each columns and rows through the dataset get the column label and row values
  print("Feature Name :", col, "        Data Type :", df[col].dtype, "\n")       # print column lable, its datatype and a new line
  print(df[col].unique(), "\n")         # print unique values
  print("_____________________________________________________________________________________")
```

output

Feature Name : AccountID Data Type : int64

[20000 20001 20002 ... 31257 31258 31259]

_____

Feature Name : Churn Data Type : int64

[1 0]

_____

Feature Name : Tenure Data Type : object

[4 0 2 13 11 '#' 9 99 19 20 14 8 26 18 5 30 7 1 23 3 29 6 28 24 25 16 10

 15 22 nan 27 12 21 17 50 60 31 51 61]

_____

Feature Name : City_Tier Data Type : float64

[3. 1. nan 2.]

Feature Name : CC_Contacted_LY Data Type : float64

[6. 8. 30. 15. 12. 22. 11. 9. 31. 18. 13. 20. 29. 28.
26. 14. 10. 25. 27. 17. 23. 33. 19. 35. 24. 16. 32. 21.
nan 34. 5. 4. 126. 7. 36. 127. 42. 38. 37. 39. 40. 41.
132. 43. 129.]

Feature Name : Payment Data Type : object

['Debit Card' 'UPI' 'Credit Card' 'Cash on Delivery' 'E wallet' nan]

Feature Name : Gender Data Type : object

['Female' 'Male' 'F' nan 'M']

Feature Name : Service_Score Data Type : float64

[3. 2. 1. nan 0. 4. 5.]

Feature Name : Account_user_count Data Type : object

[3 4 nan 5 2 '@' 1 6]

Feature Name : account_segment Data Type : object

['Super' 'Regular Plus' 'Regular' 'HNI' 'Regular +' nan 'Super Plus'
'Super +']

Feature Name : CC_Agent_Score Data Type : float64

[2. 3. 5. 4. nan 1.]

Feature Name : Marital_Status Data Type : object

['Single' 'Divorced' 'Married' nan]

Feature Name : rev_per_month Data Type : object

[9 7 6 8 3 2 4 10 1 5 '+' 130 nan 19 139 102 120 138 127 123 124 116 21
126 134 113 114 108 140 133 129 107 118 11 105 20 119 121 137 110 22 101
136 125 14 13 12 115 23 122 117 131 104 15 25 135 111 109 100 103]

Feature Name : Complain_ly Data Type : float64

[1. 0. nan]

Feature Name : rev_growth_yoy Data Type : object

[11 15 14 23 22 16 12 13 17 18 24 19 20 21 25 26 '$' 4 27 28]

Feature Name : coupon_used_for_payment Data Type : object

[1 0 4 2 9 6 11 7 12 10 5 3 13 15 8 '#' '$' 14 '*' 16]

Feature Name : Day_Since_CC_connect Data Type : object

[5 0 3 7 2 1 8 6 4 15 nan 11 10 9 13 12 17 16 14 30 '$' 46 18 31 47]

Feature Name : cashback Data Type : object

[159.93 120.9 nan ... 227.36 226.91 191.42]

Feature Name : Login_device Data Type : object

['Mobile' 'Computer' '&&&&' nan]

_____

The above code iterate through each feature in the dataset and fetch its unique values and print the same along with its name and data type. While looking at the output, there are some invalid characters such as #, @, *, $ and & present in some features.Also found some inconsistancy in the data in some features such as Gender and Account segments. Gender has been tracked as Female, F, Male, M. Similary, account_segment Regular Plus, Regular +, Super Plus and Super + . These irrgularities need to be corrected in the Feature Engineering section.

Missing value identification (DU5)

---

[267]

0s

```
df.isnull().any() #returns if there is any null values in each feature
```

output

```
AccountID              False
Churn                  False
Tenure                 True
City_Tier              True
CC_Contacted_LY        True
Payment                True
Gender                 True
Service_Score          True
Account_user_count     True
account_segment        True
CC_Agent_Score         True
Marital_Status         True
rev_per_month          True
Complain_ly            True
rev_growth_yoy         False
```

coupon_used_for_payment False

Day_Since_CC_connect True

cashback True

Login_device True

dtype: bool

There are null values almost all the fields except AccountID, Churn, rev_growth_yoy and coupon_used_for_payment.

---

[268]

0s

```
df.isnull().sum()
```

output

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

Percentage of Missing Values (DU6)

---

[269]

1s

```
# missing value percenage ploting-- Adarsh
missing = pd.DataFrame((df.isnull().sum() * 100 / df.shape[0]).reset_index())
plt.figure(figsize=(16, 5))
ax = sns.pointplot(x='index', y=0, data=missing)
plt.xticks(rotation=90, fontsize=7)
plt.title("Percentage of Missing values")
plt.ylabel("PERCENTAGE")
plt.show()
```

output

Duplicate value check (DU7)

---

[270]

0s

```
df[df.duplicated()]     # check for any duplicate values
```

output

---

[271]

0s

```
df.describe().T
```

output

The function Dataframe.duplicated() returned only the column headers. it means there are no duplicate rows\values in the dataset

Data understanding Summary (DU8)

There are 19 columns/Features 11260 records in the dataset

The presence of invalid characters in numerical fields makes the numerical data type to object data types.

There are presence of null values in most of the fields.

There are no duplicate rows in the dataset

Feature Engineering

Feature engineering includes Feature Removal, Data Cleaning, Null value treatmentand Outlier Detection/Removal.

1. Feature Removal (FE1)

[272]

0s

```
# Feature removal
print("Column counts before removal:", len(df.columns))
df.drop("AccountID", inplace=True, axis= 1)
print("Column counts after removal:", len(df.columns))
```

output

Column counts before removal: 19

Column counts after removal: 18

Double-click (or enter) to edit

[273]

0s

```
df.shape
```

output

(11260, 18)

AccountID is a unique identification number given for each rows. It doesn't add value, so the AccountId field is removing. After removing the AccountID, we have 18 features.

---

2. Data Cleaning (FE2)

While analysing the data, there were the presence of invalid characters and inconsistancies in the dataset . The below codes will remove all the invalid characters from the dataset.

---

[274]

0s

```
# Cleaning special characters
df.replace(to_replace=['*','&', '$', '+', '@','#'], value=np.NaN, inplace=True)
df.replace(to_replace='&&&&', value='others', inplace=True)

# Correcting the inconsistent data
df['Gender'].replace(to_replace='Female', value='F', inplace=True)
df['Gender'].replace(to_replace='Male', value='M', inplace=True)
df['account_segment'].replace(to_replace='Regular +', value='Regular Plus', inplace=True)
df['account_segment'].replace(to_replace='Super +', value='Super Plus', inplace=True)
```

---

[275]

0s

```
# To identify the unique values in the dataset
for col, ro in df.items():          # iterate each columns and rows through the dataset get the column label an
d row values
  print("Feature Name :", col, "        Data Type :", df[col].dtype, "\n")       # print column lable, its datatyp
e and a new line
  print(df[col].unique(), "\n")         # print unique values
  print("_____________________________________________________________________________________")

df.info()
```

output

Feature Name : Churn   Data Type : int64

[1 0]

Feature Name : Tenure Data Type : float64

[4. 0. 2. 13. 11. nan 9. 99. 19. 20. 14. 8. 26. 18. 5. 30. 7. 1.
23. 3. 29. 6. 28. 24. 25. 16. 10. 15. 22. 27. 12. 21. 17. 50. 60. 31.
51. 61.]

Feature Name : City_Tier Data Type : float64

[3. 1. nan 2.]

Feature Name : CC_Contacted_LY Data Type : float64

[6. 8. 30. 15. 12. 22. 11. 9. 31. 18. 13. 20. 29. 28.
26. 14. 10. 25. 27. 17. 23. 33. 19. 35. 24. 16. 32. 21.
nan 34. 5. 4. 126. 7. 36. 127. 42. 38. 37. 39. 40. 41.
132. 43. 129.]

Feature Name : Payment Data Type : object

['Debit Card' 'UPI' 'Credit Card' 'Cash on Delivery' 'E wallet' nan]

Feature Name : Gender Data Type : object

['F' 'M' nan]

Feature Name : Service_Score Data Type : float64

[3. 2. 1. nan 0. 4. 5.]

Feature Name : Account_user_count Data Type : float64

[3. 4. nan 5. 2. 1. 6.]

Feature Name : account_segment Data Type : object

['Super' 'Regular Plus' 'Regular' 'HNI' nan 'Super Plus']

Feature Name : CC_Agent_Score Data Type : float64

[2. 3. 5. 4. nan 1.]

Feature Name : Marital_Status Data Type : object

['Single' 'Divorced' 'Married' nan]

Feature Name : rev_per_month Data Type : float64

[9. 7. 6. 8. 3. 2. 4. 10. 1. 5. nan 130. 19. 139.
102. 120. 138. 127. 123. 124. 116. 21. 126. 134. 113. 114. 108. 140.
133. 129. 107. 118. 11. 105. 20. 119. 121. 137. 110. 22. 101. 136.
125. 14. 13. 12. 115. 23. 122. 117. 131. 104. 15. 25. 135. 111.
109. 100. 103.]

Feature Name : Complain_ly Data Type : float64

[1. 0. nan]

Feature Name : rev_growth_yoy Data Type : float64

[11. 15. 14. 23. 22. 16. 12. 13. 17. 18. 24. 19. 20. 21. 25. 26. nan 4.
27. 28.]

Feature Name : coupon_used_for_payment Data Type : float64

[1. 0. 4. 2. 9. 6. 11. 7. 12. 10. 5. 3. 13. 15. 8. nan 14. 16.]

Feature Name : Day_Since_CC_connect Data Type : float64

[5. 0. 3. 7. 2. 1. 8. 6. 4. 15. nan 11. 10. 9. 13. 12. 17. 16.

14. 30. 46. 18. 31. 47.]

Feature Name : cashback Data Type : float64

[159.93 120.9 nan ... 227.36 226.91 191.42]

Feature Name : Login_device Data Type : object

['Mobile' 'Computer' 'others' nan]

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 11260 entries, 0 to 11259

Data columns (total 18 columns):

#	Column	Non-Null Count	Dtype
0	Churn	11260 non-null	int64
1	Tenure	11042 non-null	float64
2	City_Tier	11148 non-null	float64
3	CC_Contacted_LY	11158 non-null	float64
4	Payment	11151 non-null	object
5	Gender	11152 non-null	object
6	Service_Score	11162 non-null	float64
7	Account_user_count	10816 non-null	float64
8	account_segment	11163 non-null	object
9	CC_Agent_Score	11144 non-null	float64

```
 10  Marital_Status          11048 non-null  object
 11  rev_per_month           10469 non-null  float64
 12  Complain_ly             10903 non-null  float64
 13  rev_growth_yoy          11257 non-null  float64
 14  coupon_used_for_payment  11257 non-null  float64
 15  Day_Since_CC_connect     10902 non-null  float64
 16  cashback                10787 non-null  float64
 17  Login_device            11039 non-null  object
dtypes: float64(12), int64(1), object(5)
memory usage: 1.5+ MB
```

All the invalid characters has been removed. Atfer removing the invalid characters. we have a clean datasaet of numerical and categorical fields.

---

3. Feature name correction (FE3)

For the purpose of clarity in the Feature names few labels are renamed for better understanding as below

---

[276]

0s

```
column_mapping = {
   'CC_Contacted_LY': 'Contacted_CC_in_lst_12M',
   'Service_Score': 'Service_Rating',
   'CC_Agent_Score' : 'Customer_CC_Rating',
   'rev_per_month' : 'Avg_Revenue_per_Mnth',
   'Complain_ly' : 'Complaint_recd_L_Yr',
   'rev_growth_yoy' : 'Percent_Annual_rev_growth'
   }
df=df.rename(columns=column_mapping)
print(df.columns)
df.info()
```

output

```
Index(['Churn', 'Tenure', 'City_Tier', 'Contacted_CC_in_lst_12M', 'Payment',
```

```
      'Gender', 'Service_Rating', 'Account_user_count', 'account_segment',
      'Customer_CC_Rating', 'Marital_Status', 'Avg_Revenue_per_Mnth',
      'Complaint_recd_L_Yr', 'Percent_Annual_rev_growth',
      'coupon_used_for_payment', 'Day_Since_CC_connect', 'cashback',
      'Login_device'],
     dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11260 entries, 0 to 11259
Data columns (total 18 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Churn                  11260 non-null  int64
 1   Tenure                 11042 non-null  float64
 2   City_Tier              11148 non-null  float64
 3   Contacted_CC_in_lst_12M    11158 non-null  float64
 4   Payment                11151 non-null  object
 5   Gender                 11152 non-null  object
 6   Service_Rating         11162 non-null  float64
 7   Account_user_count     10816 non-null  float64
 8   account_segment        11163 non-null  object
 9   Customer_CC_Rating     11144 non-null  float64
 10  Marital_Status         11048 non-null  object
 11  Avg_Revenue_per_Mnth   10469 non-null  float64
 12  Complaint_recd_L_Yr    10903 non-null  float64
 13  Percent_Annual_rev_growth  11257 non-null  float64
 14  coupon_used_for_payment    11257 non-null  float64
 15  Day_Since_CC_connect   10902 non-null  float64
 16  cashback               10787 non-null  float64
```

```
 17  Login_device            11039 non-null  object
dtypes: float64(12), int64(1), object(5)
memory usage: 1.5+ MB
```

The below code with create lists of categorical (cat_fld) and numerical fields (num_fld) for target operations.

---

[277]

0s

```
cat_fld=[]
num_fld=[]
for col_s in df.columns:
  if (df[col_s].dtype =='O'):
    cat_fld.append(col_s)
  else:
    num_fld.append(col_s)

# Eventhough some features are numerical datatype but they are categorical
num_fld.remove('City_Tier')
num_fld.remove('Service_Rating')
num_fld.remove('Account_user_count')
num_fld.remove('Customer_CC_Rating')
num_fld.remove('Complaint_recd_L_Yr')
num_fld.remove('Churn')

cat_fld.append('City_Tier')
cat_fld.append('Service_Rating')
cat_fld.append('Account_user_count')
cat_fld.append('Customer_CC_Rating')
cat_fld.append('Complaint_recd_L_Yr')
cat_fld.append('Churn')


print("Categorical fields:")
print("------------------")
for i in cat_fld:
  print(i)

print("\n Numerical fields:")
print("------------------")
for i in num_fld:
  print(i)
```

output

Categorical fields:

Payment

Gender

account_segment

Marital_Status

Login_device

City_Tier

Service_Rating

Account_user_count

Customer_CC_Rating

Complaint_recd_L_Yr

Churn

Numerical fields:

Tenure

Contacted_CC_in_lst_12M

Avg_Revenue_per_Mnth

Percent_Annual_rev_growth

coupon_used_for_payment

Day_Since_CC_connect

cashback

---

4. Missing value treatment (FE4)

---

[278]

0s

```
# missing value treatment for categorical field
for itms in cat_fld:
  df[itms].fillna(df[itms].mode()[0], inplace = True)

# missing value treatment for numerical field
for itms in num_fld:
  df[itms].fillna(round(df[itms].mean(),0), inplace = True)


print("Categorical fields")
print("_____________________")
print(df[cat_fld].isna().sum())
print(" ")
print("Numerical fields")
print("_____________________")
print(df[num_fld].isna().sum())
```

output

Categorical fields

_____

Payment 0

Gender 0

account_segment 0

Marital_Status 0

Login_device 0

City_Tier 0

Service_Rating 0

Account_user_count 0

Customer_CC_Rating 0

Complaint_recd_L_Yr 0

Churn 0

dtype: int64

Numerical fields

_____

Tenure 0

Contacted_CC_in_lst_12M 0

Avg_Revenue_per_Mnth 0

Percent_Annual_rev_growth 0

coupon_used_for_payment 0

Day_Since_CC_connect 0

cashback 0

dtype: int64

All the null values in numerical fields are replaced with mean and categorical fileds are replaced with mode.

---

5. Outlier detection and removal. (FE5)

---

[279]

1s

```
f, ax = plt.subplots(figsize=(10,5))
ax = sns.boxplot(data=df[num_fld],width=0.6,palette="Set3",orient='h',linewidth=0.5)
```

output

Outliers are the extereme values present in the numerical fields which All the numerical fields has outliers. Cashback and Avg_Revenue_Per_Mnth has most numebers or outliers.

---

[280]

0s

```
def get_limits(col):        #Function to get upper limit and lower limit of 5 number summary
    sorted(col)
    Q1,Q3=np.percentile(col,[25,75])
    IQR=Q3-Q1
    l_lmt= Q1-(1.5 * IQR)
    u_lmt= Q3+(1.5 * IQR)
    return u_lmt, l_lmt

#Outlier Treatment for the numeric columns
```

```
for itm in num_fld:
  ul,ll=get_limits(df[itm])
  df[itm]=np.where(df[itm]<ll,ll,df[itm])
  df[itm]=np.where(df[itm]>ul, ul,df[itm])
```

-The above code get the Upper limit and lower limit values in the data

---

Outlier detection and removal. (FE6)

---

[281]

0s

```
f, ax = plt.subplots(figsize=(10,5))
ax = sns.boxplot(data=df[num_fld],width=0.6,palette="Set3",orient='h',linewidth=0.5)
```

output

Now the data is cleaned, removed inconsistancies and outliers. The data is now ready to go ahead with EDA

---

[282]

0s

```
df.describe().T
```

output

Exploratory Data Analysis

---

1. Univariate Analysis (EDA1)

---

[283]

1s

```
percent=(df.value_counts('Churn').sort_index()/df['Churn'].count()) # calculating the Percentage of cuurn <item count/total>
fig, axs = plt.subplots(1, 2, figsize=(15,5))                    #defining subplot 15" * 5"

fig.tight_layout()                                  #to auto adjust the padding between plots
```

```
axs[0].set_title("Churn count")                          #setting title for fig 1 (countplot)
sns.countplot(x=df['Churn'], hue=df['Churn'],  ax=axs[0])        #ploting the bar grapht for each item

axs[1].pie(percent, labels=list(round(percent,2)))            #ploting percentage of each item in pie chart
axs[1].set_title("Churn Percentage")                      #setting title for fig 2 (dougnut plot)
c = plt.Circle( (0,0), 0.7, color='white')                #adding white circle to overlay pie plot
p=plt.gcf()
p.gca().add_artist(c)
plt.show()
```

output

Among all the users (11260) in the company about 17% of the users churned

---

Numerical fields (EDA2)

---

[284]

3s

```
#distribution numerical fields
fig, ax = plt.subplots(4, 2, figsize=(30, 20))              # defining subplot  4*2
for i, subplot in zip(num_fld, ax.flatten()):               # iterating through num_flds, subplot axis <ax.flatt
ern convert the 2d array to 1d to iterate>
    sns.histplot(df[i], kde = True, ax=subplot)              # ploting each histplot in numerical fileds to subpl
ot
```

output

From the above bar chart it is observed that:

Majority of Customers have a tenure in the range of 1-20 Months.

Majority of customers contacted the Customer Care about 5-25 times in the last 12 months

Average revenue growth per customer is about 15

Average Cashback obtained by customer is about 150

Categorical fields (EDA3)

---

[285]

1s

```
fig,ax = plt.subplots(nrows=4,ncols=3,figsize=(30,25))
plt.rcParams['font.size'] = '16'
for col,sub_plt in zip(cat_fld, ax.flatten()):
  sns.barplot(x=df[col].value_counts().values, y=df[col].value_counts().index,orient="h",  ax=sub_plt).set(
title=col)
plt.show()
```

output

From the above bar charts it is observed that:

Majority of users are choosing Debit card and Credit Card as their payment mode.

Significatly more no. of Male Users are there when compared to Female Users

Among all the Account segments available most of the Users belong to the "Regular Plus" and "Super"

about 50% of the users are married.

Large portion of the users are using "Mobile" as their login device

About 99% of the users belong to Tire1 and Tire3 Cities

Majority of Users rated the service score as 3/5 which signifies "AVerage"

Majority of Users accounts are being used by 4 members

Bi-variate analsysis (EDA4)

Numerical Field Analysis (EDA4)

---

[286]

3s

```
#Churn rate by category
fig,ax = plt.subplots(nrows=4,ncols=3,figsize=(30,20))
plt.rcParams['font.size'] = '16'
for col,sub_plt in zip(cat_fld, ax.flatten()):
   sns.countplot(x = df[col],  hue=df.Churn,ax=sub_plt, palette = "Greens")
plt.show()
```

output

The above plot shows the details of the Churned users:

Majority of the Churned users belong to the account segment of "Regular Plus" and "Super"

Majority of Churned users used Mobile as their login device

Majority of Churned users have given service score as 3/5

Majority of Churned user accounts has 4 users per account

Categorical variable (EDA5)

---

[287]

0s

```
fig, ax = plt.subplots(2, 3, figsize=(30, 20))
plt.rcParams['font.size'] = '18'                        # set the font size to 18
ax[0, 0].set_title("Percentage Churn Rate by Category fields")  # Add title to the first graph

for col, subplot in zip(cat_fld, ax.flatten()):          # loop to iterate over category fields
    sm = df.groupby(by=df[col]).Churn.sum()              # Calculate percent churn
    tot = df.value_counts(col).sort_index()              # Calculate total number of customers in each categor
y
    pers = sm/tot*100                                    # Calculate percent churn for each category
    subplot.set_title(f"Churn Rate by {col}")            # Add title to the graph
    wedges, texts, autotexts = subplot.pie(pers.values, labels=pers.index, autopct='%.0f%%', textprops={'fo
ntsize': 16})  # Add pie chart
    center_circle = plt.Circle((0, 0), 0.7, color='white')  # Add a white circle in the center to convert to a do
ughnut chart
    subplot.add_artist(center_circle)

plt.show()
```

output

The above plot represents few observations about the ***Churned users***:

Even though all the payment modes are used by users the maximum users used Cash on Deivery as their payment mode (26%)

Majority belong to account segment "Regular Plus"(41%) and "HNI" (24%)

About 51% has their Marital Status as "Single"

The login devices used by churned users are Computer (56%) and Mobile(44%)

Tire1 City(26%), Tire2 City(36%), Tire3 City(38%)

Multivariate Analysis (EDA6)

[288]

1s

```
msk=np.zeros_like(df.corr())                    #create a 2d arrays of zeros for each feature in datafram
e
msk[np.triu_indices_from(msk)] = True                    #split the array diagonally so that we have two par
t of zeros and ones
fig, ax = plt.subplots( figsize=(30, 20))                #defining the subplot
#sns.heatmap(df.corr(method='spearman'), mask=msk,  cmap='GnBu', annot=True)    #ploting heatmap
sns.heatmap(df.corr(method='pearson'), mask=msk,  cmap='GnBu', annot=True)    #ploting heatmap
plt.show()

# triu_indices_from: reference <https://numpy.org/doc/stable/reference/generated/numpy.triu_indices_fro
m.html>
# zeros_like: https://numpy.org/doc/stable/reference/generated/numpy.zeros_like.html
```

output

The above correlation Plot established:

Positive Correlation between:

Churn - Complain_ly (0.25)

Tenure - Cashback (0.42)

Servic_Score - Account_User_count (0.32)

Coupon_Used_for_payment - Day_Since_CC_connect (0.35)

Day_Since_CC_Connect - (0.34)

Negative Correlation between:

Churn - Tenure (-0.33)

Churn - Day_Since_CC_Connect (-0.15)

Churn - Cashback (-0.15)

Machine Learning (Classification)

Importing ML Libraries

Data Pre-Processing

Model Building

Model Evaluation

1. Importing ML Libraries (ML1)

[289]

0s

```
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, roc_auc_score, classification_report, RocCurveDisplay
from sklearn.metrics import f1_score
from sklearn.preprocessing import StandardScaler


# importing classifiers for model building
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from xgboost import XGBClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import  GridSearchCV


import warnings
warnings.filterwarnings("ignore")


from mlxtend.plotting import plot_confusion_matrix
```

Data Pre-Processing (ML2)

creating dataset copy

[290]

0s

```
churn_data=df  # Creating dataset for ML operations
churn_data.shape
```

output

(11260, 18)

Label encoding (ML2A)

---

[291]

0s

```
from sklearn.preprocessing import LabelEncoder
# creating instance of labelencoder
labelencoder = LabelEncoder()
# Assigning numerical values and storing in another column
churn_data['Payment'] = labelencoder.fit_transform(churn_data['Payment'])
churn_data['Gender'] = labelencoder.fit_transform(churn_data['Gender'])
churn_data['account_segment'] = labelencoder.fit_transform(churn_data['account_segment'])
churn_data['Marital_Status'] = labelencoder.fit_transform(churn_data['Marital_Status'])
churn_data['Login_device'] = labelencoder.fit_transform(churn_data['Login_device'])
churn_data.head(3)
```

output

Dataset Splitting Y - target, X - Predictors (ML2B)

---

[292]

0s

```
x=churn_data.drop("Churn", axis=1)
y=churn_data.pop("Churn")
```

---

[293]

0s

```
x.shape
```

output

(11260, 17)

---

[294]

0s

```
y.shape
```

output

(11260,)

Feature Standardization (ML3)

---

[295]

0s

```
x.head()
```

output

---

[296]

0s

```
from scipy.stats import zscore
```

---

[297]

0s

```
# Applying Z score
x=x.apply(zscore)
x.head()
```

output

Train Test split 70:30 (ML4)

---

[298]

0s

```
x_train, x_test,y_train,y_test = train_test_split(x,y,test_size=.30,random_state=1)
```

---

[299]

0s

```
x_train.shape
```

output

(7882, 17)

---

[300]

0s

x_test.shape

output

(3378, 17)

Model Building (Model : LDA) (ML-5)

---

[301]

0s

```
model_lda = LinearDiscriminantAnalysis()
model_LogisReg = LogisticRegression()
model_knn = KNeighborsClassifier()
model_xgb = XGBClassifier()
model_svm = SVC()
model_RandFrst = RandomForestClassifier()
```

---

[302]

0s

```
def get_performance_score(actual, prediction):
  tn, fp, fn, tp= metrics.confusion_matrix(actual,prediction).ravel()
  accuracy = round((tp + tn)/ (tp + fn + tn + fp),3)  # compute accuracy score
  recall=round(tp/(tp+fn), 3)                         # compute recall score
  precision= round(tp/(tp+fp),3)                      # compute precision score
  specificity= round(tn/(tn+fp), 3)                   # compute specificity
  fi_score = round(2*precision*recall/(precision+recall),3)  # compute f1_score
  return  accuracy, recall, precision, specificity, fi_score
```

---

[303]

0s

```
models=[model_lda , model_LogisReg, model_knn, model_xgb, model_svm, model_RandFrst ]
#models= [model_lda, model_LogisReg , model_svm , model_DisTree , model_RandFrst, model_xgb]
model_names= ['Linear discriminant analysis','Logistic Regression', 'K-
Nearest Neighbors','XGBClassifier', 'Support Vector Machine', 'Random Forest' ]
```

---

[304]

0s

```
cols = ['Model', 'accuracy', 'recall', 'precision',      'specificity',      'f1_score']
test_scores = pd.DataFrame(columns = cols)
train_scores = pd.DataFrame(columns = cols)
```

---

[305]

12s

```
test_scores.drop(test_scores.index , inplace=True)
train_scores.drop(train_scores.index, inplace=True)
for model,name in zip(models,model_names):
  model.fit(x_train,y_train)                                                    # fit the model using the tra
in data
  #----------------------------------------
  y_pred_train = model.predict(x_train)                                          # Predicting Y using t
rain data for comparison
  y_pred_test = model.predict(x_test)                                           # Predicting Y using te
st data
  #----------------------------------------
  # functios applied on test and train data to return accuracy, recall, precision, specifiity and f1 score
  tst_acc, tst_rcll, tst_prec, tst_speci, tst_f1Score= get_performance_score(y_test,y_pred_test)
  tst_modelScore=round(model.score(x_test , y_test),3)                                  # getting mode
l scores for test
  trn_acc, trn_rcll, trn_prec, trn_speci, trn_f1Score= get_performance_score(y_train,y_pred_train)
  trn_modelScore=round(model.score(x_train, y_train), 3)                                 # getting mod
el scores for train dataset

  # Adding performace scores for test and train to different dataset for evaluation
  test_scores = test_scores.append({'Model':name, 'accuracy': tst_acc, 'recall':tst_rcll, 'precision':tst_prec,'s
pecificity':tst_speci,      'f1_score':tst_f1Score  },ignore_index=True)
  train_scores = train_scores.append({'Model':name, 'accuracy': trn_acc, 'recall':trn_rcll, 'precision':trn_pre
c,'specificity':trn_speci,    'f1_score':trn_f1Score  },ignore_index=True)
  # Printing performance score for each model
  print(f'Using model: {name}')
  print("               ", "mdl_score","\taccuracy \t recall", "\t precision", "\t specificity","\t f1_score")
  print("Trainning Score: ",  "{:.3f}".format(trn_modelScore) ,"\t", "{:.3f}".format(trn_acc),"\t\t", "{:.3f}".
format(trn_rcll), "\t\t","{:.3f}".format(trn_prec),"\t\t", "{:.3f}".format(trn_speci),"\t\t", "{:.3f}".format(trn_
```

```
f1Score))
  print(f"Test Score:      ", "{:.3f}".format(tst_modelScore), "\t" ,"{:.3f}".format(tst_acc), "\t\t","{:.3f}".for
mat(tst_rcll), "\t\t" ,"{:.3f}".format(tst_prec), "\t\t","{:.3f}".format(tst_speci), "\t\t","{:.3f}".format(tst_f1S
core))
  print("_____
_____")
```

output

Using model: Linear discriminant analysis

	mdl_score	accuracy	recall	precision	specificity	f1_score
Trainning Score:	0.875	0.875	0.394	0.743	0.972	0.515
Test Score:	0.881	0.881	0.412	0.776	0.976	0.538

_____
_____

Using model: Logistic Regression

	mdl_score	accuracy	recall	precision	specificity	f1_score
Trainning Score:	0.882	0.882	0.446	0.754	0.971	0.560
Test Score:	0.888	0.888	0.460	0.787	0.975	0.581

_____
_____

Using model: K-Nearest Neighbors

	mdl_score	accuracy	recall	precision	specificity	f1_score
Trainning Score:	0.978	0.978	0.895	0.971	0.995	0.931
Test Score:	0.957	0.957	0.807	0.931	0.988	0.865

_____
_____

Using model: XGBClassifier

	mdl_score	accuracy	recall	precision	specificity	f1_score
Trainning Score:	0.999	0.999	0.997	1.000	1.000	0.998
Test Score:	0.971	0.971	0.868	0.954	0.991	0.909

_____
_____

Using model: Support Vector Machine

	mdl_score	accuracy	recall	precision	specificity	f1_score
Trainning Score:	0.944	0.944	0.710	0.945	0.992	0.811
Test Score:	0.928	0.928	0.616	0.936	0.991	0.743

Using model: Random Forest

	mdl_score	accuracy	recall	precision	specificity	f1_score
Trainning Score:	1.000	1.000	1.000	1.000	1.000	1.000
Test Score:	0.972	0.972	0.851	0.982	0.997	0.912

Double-click (or enter) to edit

Model Evaluation (MLEV)

---

[306]

0s

```
df_model_acc= pd.DataFrame()
df_model_acc['Models'] = train_scores['Model']
df_model_acc['Train Accuracy']= round(train_scores['accuracy'],3)
df_model_acc['Test Accuracy']= round(test_scores['accuracy'],3)
df_model_acc.set_index('Models', inplace=True)
df_model_acc
```

output

---

[307]

0s

```
ax = df_model_acc.plot.barh(title="Model Acccuracy Comparison",
          figsize=(20, 10))
for container in ax.containers:
   ax.bar_label(container)
```

```
ax.legend(loc=4, fontsize=12)
ax.set_xlabel("Accuracy")
plt.show()
```

output

From the above Graph it is observed that Random Forest, Support Vector Machine(SVM), XGB Classifier, KNN are having higher Test Accuracy compared to Train Accuracy, Whereas Logistic Regression and Linear Discriminant analysis (LDA) are having Train Accuracy and Test Accuracy in line.

---

[308]

0s

```
train_scores
```

output

---

[309]

0s

```
test_scores
```

output

From the above Trained and Test Scores it is observed that

Accuracy is good For all Models with above 90% accuracy and precision for KNN, XGB Classifier, Support Vector Machine (SVM) and Random Forest.

All the models having the Specificity over 95%.

KNN, XGB Classifier, Random Forest has higher Recall which is over 80%

The F1 score for Random Forest is 91.4% followed by XGB Classifier with 90.9% followed by KNN with 86.5%.

As XGB Classifier is the developed version of Random Forest Method. The XGB classifier ML Algorithm is recommended as a Model for predecting the present problem i.e., Customer Churn.

Double-click (or enter) to edit

Double-click (or enter) to edit

[309]

0s

Model Evaluation KNN(MLEV1)

[310]

1s

```
rep_df= pd.DataFrame() #Creating dataframe to store test train model performance scores
# Model: K-Nearest Neighbours classfier
model_name="K-Nearest Neighbours classfier"
model= KNeighborsClassifier()                                  # calling the model classifier
model.fit(x_train,y_train)
y_pred_tr = model.predict(x_train)
y_pred_ts = model.predict(x_test)                                 # Predicting the y usuing test data
# accuracy = accuracy_score(y_test, y_pred)
roc_auc1_ts = roc_auc_score(y_test, y_pred_ts)
roc_auc1_tr = roc_auc_score(y_train, y_pred_tr)

tst_acc, tst_rcll, tst_prec, tst_speci, tst_f1Score= get_performance_score(y_test,y_pred_ts)
trn_acc, trn_rcll, trn_prec, trn_speci, trn_f1Score= get_performance_score(y_train,y_pred_tr)


  # Adding performace scores for test and train to different dataset for evaluation
rep_df = rep_df.append({'Scores':'Test', 'accuracy':tst_acc, 'recall':tst_rcll, 'precision':tst_prec, 'specificity':
tst_speci, 'f1_score':tst_f1Score, 'AUC_score':roc_auc1_ts },ignore_index=True)
rep_df = rep_df.append({'Scores':'Train', 'accuracy':trn_acc, 'recall':trn_rcll, 'precision':trn_prec, 'specificit
y':trn_speci,'f1_score':trn_f1Score,  'AUC_score':roc_auc1_tr },ignore_index=True)


fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(30, 10))
fig.suptitle(model_name)
plt.rcParams['font.size'] = '30'

sns.heatmap(confusion_matrix(y_test, y_pred_ts), fmt="d", annot=True, ax=ax1)
ax2.set(xlabel='FPR', ylabel='TPR')

fpr_tr, tpr_tr, thresholds_tr = metrics.roc_curve(y_train,y_pred_tr)# [:,1])
fpr_ts, tpr_ts, thresholds_ts = metrics.roc_curve(y_test, y_pred_ts)# [:,1])            #  calculate roc curve
ax2.plot([0, 1], [0, 1], linestyle='--
```

```
',linewidth = '8')                                    # plot the roc curve for the model
ax2.plot(fpr_tr, tpr_tr, marker='.',linewidth = '8',label = ('Train AUC (%.3f)' % roc_auc1_tr))
ax2.plot(fpr_ts, tpr_ts, marker='.',linewidth = '8',label = ('Test AUC (%.3f)' % roc_auc1_ts))
ax2.legend(loc='lower right')
plt.show()
rep_df
```

output

Model Evaluation XGBoost(MLEV2)

---

[311]

1s

```
rep_df= pd.DataFrame() #Creating dataframe to store test train model performance scores
# Model: K-Nearest Neighbours classfier
model_name="Extreme Gradient Boosting (XGB Classfier)"
model= XGBClassifier()                                  # calling the model classifier
model.fit(x_train,y_train)
y_pred_tr = model.predict(x_train)
y_pred_ts = model.predict(x_test)                                  # Predicting the y usuing test data
# accuracy = accuracy_score(y_test, y_pred)
roc_auc1_ts = roc_auc_score(y_test, y_pred_ts)
roc_auc1_tr = roc_auc_score(y_train, y_pred_tr)

tst_acc, tst_rcll, tst_prec, tst_speci, tst_f1Score= get_performance_score(y_test,y_pred_ts)
trn_acc, trn_rcll, trn_prec, trn_speci, trn_f1Score= get_performance_score(y_train,y_pred_tr)


 # Adding performace scores for test and train to different dataset for evaluation
rep_df = rep_df.append({'Scores':'Test', 'accuracy':tst_acc, 'recall':tst_rcll, 'precision':tst_prec, 'specificity':
tst_speci, 'f1_score':tst_f1Score, 'AUC_score':roc_auc1_ts },ignore_index=True)
rep_df = rep_df.append({'Scores':'Train', 'accuracy':trn_acc, 'recall':trn_rcll, 'precision':trn_prec, 'specificit
y':trn_speci,'f1_score':trn_f1Score,  'AUC_score':roc_auc1_tr },ignore_index=True)


fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(30, 10))
fig.suptitle(model_name)
plt.rcParams['font.size'] = '30'

sns.heatmap(confusion_matrix(y_test, y_pred_ts), fmt="d", annot=True, ax=ax1)
ax2.set(xlabel='FPR', ylabel='TPR')

fpr_tr, tpr_tr, thresholds_tr = metrics.roc_curve(y_train,y_pred_tr)# [:,1])
```

```
fpr_ts, tpr_ts, thresholds_ts = metrics.roc_curve(y_test, y_pred_ts)# [:,1])              #  calculate roc curve
ax2.plot([0, 1], [0, 1], linestyle='--
',linewidth = '8')                                  # plot the roc curve for the model
ax2.plot(fpr_tr, tpr_tr, marker='.',linewidth = '8',label = ('Train AUC (%.3f)' % roc_auc1_tr))
ax2.plot(fpr_ts, tpr_ts, marker='.',linewidth = '8',label = ('Test AUC (%.3f)' % roc_auc1_ts))
ax2.legend(loc='lower right')
plt.show()
rep_df
```

output

Model Evaluation Support Vector Machine(MLEV3)

---

[312]

4s

```
rep_df= pd.DataFrame() #Creating dataframe to store test train model performance scores
# Model: Support Vector Machine
model_name="Support Vector Machine"
model= SVC()                                  # calling the model classifier
model.fit(x_train,y_train)
y_pred_tr = model.predict(x_train)
y_pred_ts = model.predict(x_test)                                  # Predicting the y usuing test data
# accuracy = accuracy_score(y_test, y_pred)
roc_auc1_ts = roc_auc_score(y_test, y_pred_ts)
roc_auc1_tr = roc_auc_score(y_train, y_pred_tr)

tst_acc, tst_rcll, tst_prec, tst_speci, tst_f1Score= get_performance_score(y_test,y_pred_ts)
trn_acc, trn_rcll, trn_prec, trn_speci, trn_f1Score= get_performance_score(y_train,y_pred_tr)


 # Adding performace scores for test and train to different dataset for evaluation
rep_df = rep_df.append({'Scores':'Test', 'accuracy':tst_acc, 'recall':tst_rcll, 'precision':tst_prec, 'specificity':
tst_speci, 'f1_score':tst_f1Score, 'AUC_score':roc_auc1_ts },ignore_index=True)
rep_df = rep_df.append({'Scores':'Train', 'accuracy':trn_acc, 'recall':trn_rcll, 'precision':trn_prec, 'specificit
y':trn_speci,'f1_score':trn_f1Score,  'AUC_score':roc_auc1_tr },ignore_index=True)


fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(30, 10))
fig.suptitle(model_name)
plt.rcParams['font.size'] = '30'

sns.heatmap(confusion_matrix(y_test, y_pred_ts), fmt="d", annot=True, ax=ax1)
ax2.set(xlabel='FPR', ylabel='TPR')
```

```
fpr_tr, tpr_tr, thresholds_tr = metrics.roc_curve(y_train,y_pred_tr)# [:,1])
fpr_ts, tpr_ts, thresholds_ts = metrics.roc_curve(y_test, y_pred_ts)# [:,1])            #  calculate roc curve
ax2.plot([0, 1], [0, 1], linestyle='--
',linewidth = '8')                                  # plot the roc curve for the model
ax2.plot(fpr_tr, tpr_tr, marker='.',linewidth = '8',label = ('Train AUC (%.3f)' % roc_auc1_tr))
ax2.plot(fpr_ts, tpr_ts, marker='.',linewidth = '8',label = ('Test AUC (%.3f)' % roc_auc1_ts))
ax2.legend(loc='lower right')
plt.show()
rep_df
```

output

Model Evaluation Random Forest(MLEV4)

---

[313]

2s

```
rep_df= pd.DataFrame() #Creating dataframe to store test train model performance scores
# Model: Random Forest Classfier
model_name="Random Forest Classfier"
model= RandomForestClassifier()                                  # calling the model classifier
model.fit(x_train,y_train)
y_pred_tr = model.predict(x_train)
y_pred_ts = model.predict(x_test)                                # Predicting the y usuing test data
# accuracy = accuracy_score(y_test, y_pred)
roc_auc1_ts = roc_auc_score(y_test, y_pred_ts)
roc_auc1_tr = roc_auc_score(y_train, y_pred_tr)

tst_acc, tst_rcll, tst_prec, tst_speci, tst_f1Score= get_performance_score(y_test,y_pred_ts)
trn_acc, trn_rcll, trn_prec, trn_speci, trn_f1Score= get_performance_score(y_train,y_pred_tr)


 # Adding performace scores for test and train to different dataset for evaluation
rep_df = rep_df.append({'Scores':'Test', 'accuracy':tst_acc, 'recall':tst_rcll, 'precision':tst_prec, 'specificity':
tst_speci, 'f1_score':tst_f1Score, 'AUC_score':roc_auc1_ts },ignore_index=True)
rep_df = rep_df.append({'Scores':'Train', 'accuracy':trn_acc, 'recall':trn_rcll, 'precision':trn_prec, 'specificit
y':trn_speci,'f1_score':trn_f1Score,  'AUC_score':roc_auc1_tr },ignore_index=True)


fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(30, 10))
fig.suptitle(model_name)
plt.rcParams['font.size'] = '30'
```

```
sns.heatmap(confusion_matrix(y_test, y_pred_ts), fmt="d", annot=True, ax=ax1)
ax2.set(xlabel='FPR', ylabel='TPR')

fpr_tr, tpr_tr, thresholds_tr = metrics.roc_curve(y_train,y_pred_tr)# [:,1])
fpr_ts, tpr_ts, thresholds_ts = metrics.roc_curve(y_test, y_pred_ts)# [:,1])            #  calculate roc curve
ax2.plot([0, 1], [0, 1], linestyle='--
',linewidth = '8')                                  # plot the roc curve for the model
ax2.plot(fpr_tr, tpr_tr, marker='.',linewidth = '8',label = ('Train AUC (%.3f)' % roc_auc1_tr))
ax2.plot(fpr_ts, tpr_ts, marker='.',linewidth = '8',label = ('Test AUC (%.3f)' % roc_auc1_ts))
ax2.legend(loc='lower right')
plt.show()
rep_df
```

output

Cross Validation (MLCV)

---

[314]

2s

```
from sklearn.model_selection import cross_val_score, StratifiedKFold
from sklearn.metrics import roc_auc_score

xgb_model = XGBClassifier()

# Perform Stratified k-Fold Cross-Validation
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=1)
scores = cross_val_score(xgb_model, x, y, cv=kfold, scoring='accuracy')

# Print the cross-validation scores
print("Cross-Validation Scores:", scores)
print("Mean Accuracy:", scores.mean())
print("Standard Deviation:", scores.std())

# Calculate AUC for cross-validation
auc_scores = cross_val_score(xgb_model, x, y, cv=kfold, scoring='roc_auc')
print("AUC Scores:", auc_scores)
print("Mean AUC:", auc_scores.mean())
print("Standard Deviation of AUC:", auc_scores.std())
```

output

Cross-Validation Scores: [0.96536412 0.97646536 0.97424512 0.97424512 0.97380107]

Mean Accuracy: 0.9728241563055061

Standard Deviation: 0.003844557872453515

AUC Scores: [0.99196328 0.99338045 0.99426653 0.98879508 0.99192947]

Mean AUC: 0.9920669643244633

Standard Deviation of AUC: 0.0018600104737306292

Hyper Parameter Tuning (MLHPT)

---

[315]

0s

```
# Defining the models
models = {
   'KNN': KNeighborsClassifier(),
   'RF': RandomForestClassifier(),
   'SVM': SVC(),
   'XGB': XGBClassifier()
   }
```

Declaring parameters for the models (MLHPT1)

---

[316]

0s

```
# defining the hyper parameter for different models
params = {
   'KNN': {
      'n_neighbors': [3, 5, 7, 9],
      'weights': ['uniform', 'distance'],
      'metric': ['euclidean', 'manhattan']
      },
   'RF': {
      'n_estimators': [10, 101, 200],
      'criterion': ['gini', 'entropy'],
      'max_depth': [3, 5, 6],
      },
   'SVM': {
      #'kernel': ['poly', 'rbf', 'sigmoid'],
      'kernel': ['rbf'],
      'C': [0.1, 1, 10],
      #'gamma': ['scale', 'auto', 0.1, 0.01, 0.001, 0.0001]
```

```
        'gamma': [ 0.1, 0.01, 0.001]
        },

    'XGB': {
        'n_estimators': [100, 200, 300],
        'learning_rate': [0.01, 0.1, 0.2, 0.3],
        'max_depth': [3, 5, 7, 9],
       }
    }
```

---

[317]

0s

```
#Defining the evaluation metrics
metrics = {
    'accuracy': accuracy_score,
    'f1': f1_score,
    'roc_auc': roc_auc_score }

results = {}                # result dictonary to store the results
```

Hyper Parameter turning with different parameters (MLHPT2)

---

[318]

3m

```
for name, model in models.items():
  print(f'Tuning {name}…') # Print the model name
  # Create a GridSearchCV object
  grid = GridSearchCV(model, params[name], cv=5, scoring='roc_auc', n_jobs=-
1, verbose=1) # Number of folds =5 and scoring type is AUC ROC
  # Fit the grid on the training data
  grid.fit(x_train, y_train)

  # Predict on the test data
  y_pred = grid.predict(x_test)
  # Evaluate the performance and storing resulsts
  scores = {}
  for metric, func in metrics.items():
        scores[metric] = func(y_test, y_pred)
```

```
  # Store the results to the dictonary
  results[name] = {
    'best_params': grid.best_params_,
    'best_score': grid.best_score_,
    'test_scores': scores
}
```

output

```
Tuning KNN...

Fitting 5 folds for each of 16 candidates, totalling 80 fits

Tuning RF...

Fitting 5 folds for each of 18 candidates, totalling 90 fits

Tuning SVM...

Fitting 5 folds for each of 9 candidates, totalling 45 fits

Tuning XGB...

Fitting 5 folds for each of 48 candidates, totalling 240 fits
```

Getting best estimators and scores from the grid (MLHPT3)

---

[319]

0s

```
HPT_scores=pd.DataFrame()                    # Dataset for storing results for evaluation
for name, result in results.items():
  print(f'{name}:')
  print(f'Best parameters: {result["best_params"]}')
  print(f'Best cross-validation score: {result["best_score"]:.4f}')
  print(f'Test scores:')
  for metric, score in result['test_scores'].items():
    print(f'{metric}: {score:.4f}')
   # print()
  print('_____
__')
  HPT_scores=HPT_scores.append({"Model_name":name,
                "Best_Score":round(result["best_score"],4),
                "accuracy":round((result['test_scores']['accuracy']),4),
                "f1_score":round((result['test_scores']['f1']),4),
                "roc_auc":round((result['test_scores']['roc_auc']),4)},
               ignore_index=True)
```

output

KNN:

Best parameters: {'metric': 'manhattan', 'n_neighbors': 7, 'weights': 'distance'}

Best cross-validation score: 0.9876

Test scores:

accuracy: 0.9772

f1: 0.9297

roc_auc: 0.9436

RF:

Best parameters: {'criterion': 'entropy', 'max_depth': 6, 'n_estimators': 200}

Best cross-validation score: 0.9273

Test scores:

accuracy: 0.9082

f1: 0.6623

roc_auc: 0.7588

SVM:

Best parameters: {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'}

Best cross-validation score: 0.9842

Test scores:

accuracy: 0.9769

f1: 0.9301

roc_auc: 0.9505

XGB:

Best parameters: {'learning_rate': 0.2, 'max_depth': 9, 'n_estimators': 300}

Best cross-validation score: 0.9873

Test scores:

accuracy: 0.9790

f1: 0.9348

roc_auc: 0.9447

_____

---

[320]

0s

HPT_scores

output

Based on the evaluation results, XGB is the best model among the four. It has the second highest ROC_AUC score (0.9447) and the highest accuracy score (0.9790).The second is the KNN in terms of accuracy. While SVM has a third highest accuracy score (0.9769), its ROC_AUC score is higher (0.9505) among all the models. This suggests that XGB is better at discriminating between positive and negative cases.XGB outperforms the other models in terms of ROC_AUC, which is a more reliable metric for evaluating classification performance. Therefore, XGB is the best model which can be used for the customer churn classification . Moreover Looking at the Hyper parameter truning of the SVM, the results were significantly increased. As the AUC ROC is also very good SVM can be more suitable if we make further research.

---

[320]

0s