Customer Churn Analysis

(Capstone Project)

DEEPTI AGRAWAL

Submission Date: 08th December 2021

CONTENTS

Customer Churn Dataset:

Problem Statement:

An E Commerce company or DTH 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.

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

Introduction:

Here, we will be predicting churn rate for the DTH Company.
Customer churn is used to describe the loss of customers by a company. Generally, if a customer stops using the services of a company for a long period of time (which varies depending on the products or services of the company), such a customer is considered churned. Customer churn is also called customer attrition.

It is widely accepted that the cost of retention is lower than the cost of acquisition. With the event of interest being a service cancellation, Telco companies can more effectively manage customer retention efforts by using survival analysis to better predict at what point in time-specific customers are likely to be at risk of churning.

Customer churn prediction is regarded as one of the most popular use cases of big data by businesses. It is also called deflection probability. It involves ways in which customers that are likely to stop using certain products and services of a company are predicted based on how they use the products or services.
We have a historical customer data which we will be using for the prediction.

Based on the introduction the key challenge is to predict if an individual customer will churn or not. To accomplish that, machine learning models are trained based on 80% of the sample data. The remaining 20% are used to apply the trained models and assess their predictive power with regards to “churn / not churn”. A side question will be, which features actually drive customer churn. That information can be used to identify customer “pain points” and resolve them by providing goodies to make customers stay.

Data Dictionary:

AccountID	account unique identifier
Churn	account churn flag (Target)
Tenure	Tenure of account
City_Tier	Tier of primary customer's city
CC_Contacted_L12m	How many times all the customers of the account has contacted customer care in last 12months

Payment	Preferred Payment mode of the customers in the account
Gender	Gender of the primary customer of the account
Service_Score	Satisfaction score given by customers of the account on service provided by company
Account_user_count	Number of customers tagged with this account
account_segment	Account segmentation on the basis of spend
CC_Agent_Score	Satisfaction score given by customers of the account on customer care service provided by company
Marital_Status	Marital status of the primary customer of the account
rev_per_month	Monthly average revenue generated by account in last 12 months
Complain_l12m	Any complaints has been raised by account in last 12 months
rev_growth_yoy	revenue growth percentage of the account (last 12 months vs last 24 to 13 month)
coupon_used_l12m	How many times customers have used coupons to do the payment in last 12 months
Day_Since_CC_connect	Number of days since no customers in the account has contacted the customer care
cashback_l12m	Monthly average cashback generated by account in last 12 months
Login_device	Preferred login device of the customers in the account

The data has been loaded correctly to a data frame.

EXPLORATORY DATA ANALYSIS

Head of Dataset

	AccountID	Churn	Tenure	City_Tier	CC_Contacted_LY	Payment	Gender	Service_Score	Account_user_count	account_segment	CC_Agent_Score	Marital_
0	20000	1	4	3.0	6.0	Debit Card	Female	3.0	3	Super	2.0	
1	20001	1	0	1.0	8.0	UPI	Male	3.0	4	Regular Plus	3.0	
2	20002	1	0	1.0	30.0	Debit Card	Male	2.0	4	Regular Plus	3.0	
3	20003	1	0	3.0	15.0	Debit Card	Male	2.0	4	Super	5.0	
4	20004	1	0	1.0	12.0	Credit Card	Male	2.0	3	Regular Plus	5.0	

nent	CC_Agent_Score	Marital_Status	rev_per_month	Complain_ly	rev_growth_yoy	coupon_used_for_payment	Day_Since_CC_connect	cashback	Login_device
uper	2.0	Single	9	1.0	11	1	5	159.93	Mobile
Plus	3.0	Single	7	1.0	15	0	0	120.9	Mobile
Plus	3.0	Single	6	1.0	14	0	3	NaN	Mobile
uper	5.0	Single	8	0.0	23	0	3	134.07	Mobile
Plus	5.0	Single	3	0.0	11	1	3	129.6	Mobile

Shape of Dataset

```
## Checking the shape of the data: Number of columns and rows

Customer.shape
```

```
(11260, 19)
```

Info of Dataset

```
<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
```

In the data info, we can see many anomalies. The variable, cashback which is a currency (numeric) value is shown as an object.

```
#Treating cashback column
Customer[Customer.cashback=="$"]
```

ment	CC_Agent_Score	Marital_Status	rev_per_month	Complain_ly	rev_growth_yoy	coupon_used_for_payment	Day_Since_CC_connect	cashback	Login_device
Plus	3.0	Single	2	0.0	18	1	2	$	Mobile
ılar +	5.0	Married	+	NaN	13	0	3	$	Computer

We found that there is a special character present in the data, which makes it readable as a string object.

Similarly, Day_Since_CC_connect is a numeric variable but it is shown as an object due to presence of special characters.

```
Customer.Login_device.value_counts()
```

```
Mobile      7482
Computer    3018
&&&&         539
Name: Login_device, dtype: int64
```

Login_device has 539 values as '&&&&' whereas it can take only two values i.e. Mobile and Computer.

```
Customer.City_Tier.value_counts()
```

```
1.0    7263
3.0    3405
2.0     480
Name: City_Tier, dtype: int64
```

City_Tier is shown as a float value whereas it is a categorical variable.

```
#Treating rev_per_month column
Customer[Customer.rev_per_month=="+"]
```

ervice_Score	Account_user_count	account_segment	CC_Agent_Score	Marital_Status	rev_per_month	Complain_ly	rev_growth_yoy	coupon_used_for_payment
3.0	3	Regular +	4.0	Divorced	+	NaN	18	1
2.0	2	HNI	2.0	Married	+	NaN	16	1
3.0	3	Regular +	4.0	Divorced	+	NaN	13	0
3.0	4	Regular	3.0	Divorced	+	NaN	14	1
2.0	@	HNI	4.0	Divorced	+	0.0	16	1
...	...	...	...	...	...	...	...	...
3.0	5	Super	3.0	Single	+	NaN	13	4

rev_per_month and account_user_count should be numeric but are categorised as an object due to presence of special characters.

We have treated many of these variables by removing special characters.

```
Customer.info()
```

```
<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                   11042 non-null  float64
 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       10816 non-null  float64
 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            10469 non-null  float64
 13  Complain_ly              10903 non-null  float64
 14  rev_growth_yoy           11257 non-null  float64
 15  coupon_used_for_payment  11257 non-null  float64
 16  Day_Since_CC_connect     10902 non-null  float64
 17  cashback                 10787 non-null  float64
 18  Login_device             10500 non-null  object
dtypes: float64(12), int64(2), object(5)
memory usage: 1.6+ MB
```

The same can be observed in above table.

We will now check for missing values and treat them to prepare our data for further modelling.

```
rev_per_month              791
Login_device               760
cashback                   473
Account_user_count         444
Day_Since_CC_connect       358
Complain_ly                357
Tenure                     218
Marital_Status             212
CC_Agent_Score             116
City_Tier                  112
Payment                    109
Gender                     108
CC_Contacted_LY            102
Service_Score               98
account_segment             97
rev_growth_yoy               3
coupon_used_for_payment      3
dtype: int64
```

Descriptive Statistics

```
## Checking the summary
Customer.describe().round(2)
```

	AccountID	Tenure	CC_Contacted_LY	Account_user_count	rev_per_month	rev_growth_yoy	coupon_used_for_payment	Day_Since_CC_connect	cashba
count	11260.00	11042.00	11158.00	10816.00	10469.00	11257.00	11257.00	10902.00	10787
mean	25629.50	11.03	17.87	3.69	6.36	16.19	1.79	4.63	196
std	3250.63	12.88	8.85	1.02	11.91	3.76	1.97	3.70	178
min	20000.00	0.00	4.00	1.00	1.00	4.00	0.00	0.00	0
25%	22814.75	2.00	11.00	3.00	3.00	13.00	1.00	2.00	147
50%	25629.50	9.00	16.00	4.00	5.00	15.00	1.00	3.00	165
75%	28444.25	16.00	23.00	4.00	7.00	19.00	2.00	8.00	200
max	31259.00	99.00	132.00	6.00	140.00	28.00	16.00	47.00	1997

Checking for Duplicate Values

```
# Check for duplicate data

dups = Customer.duplicated()
print('Number of duplicate rows = %d' % (dups.sum()))

Customer[dups]
```

```
Number of duplicate rows = 0
```

AccountID	Churn	Tenure	City_Tier	CC_Contacted_LY	Payment	Gender	Service_Score	Account_user_count	account_segment	CC_Agent_Score	Marital_St

There are no duplicate values in our dataset.

We will now be replacing all NULL values in numerical columns using Median and the ones in categorical column using Mode.

Checking for Missing Values

Again checking for Null Values.

```
# Check for missing value in any column
Customer.isnull().sum()
```

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

```
Customer.shape
```

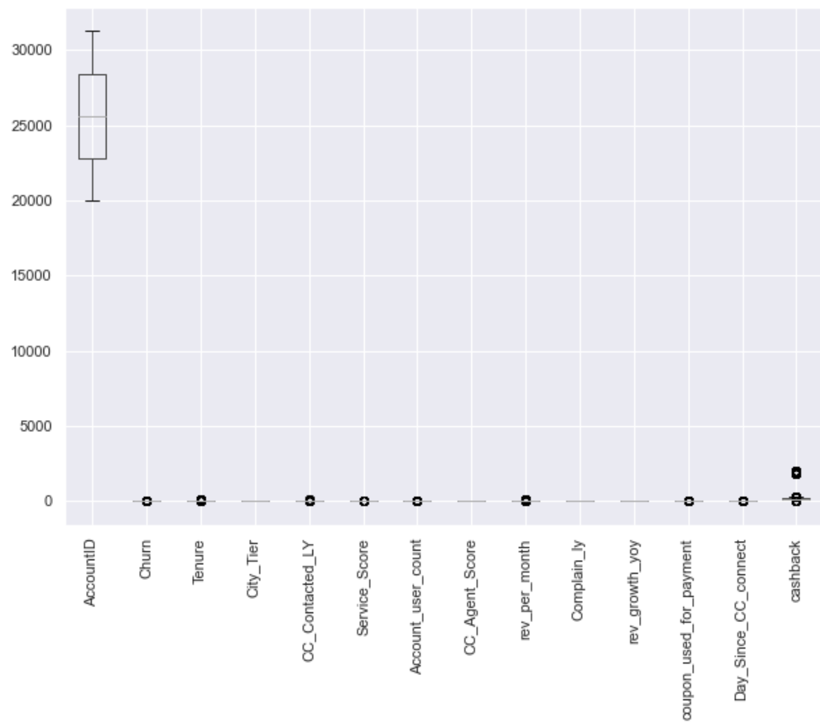
```
(11260, 19)
```

We still have same 11260 observations and 19 columns.

Checking for Outliers

Using boxplots we will check for outliers. Boxplots identify outliers based on the five point summary (min, 25%, 50%, 75%, max). Outliers are the values which fall outside the min-max range values.

```
Customer.boxplot(figsize=(10,7), rot=90);
```

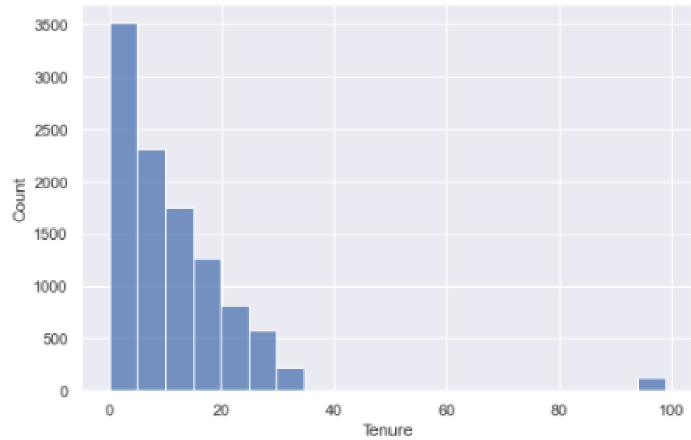


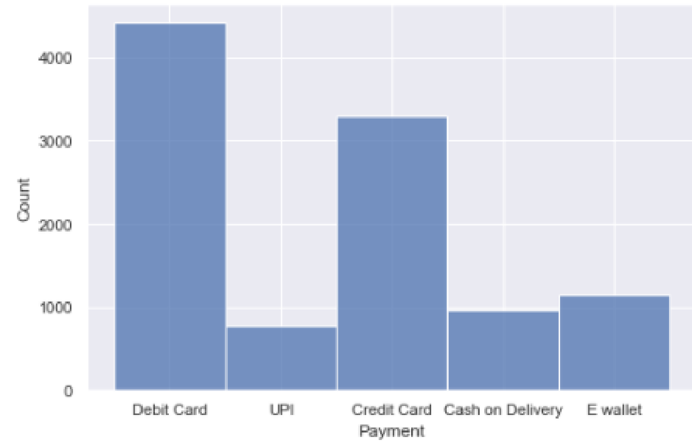
There are not many outliers present in our dataset thus treating them is not necessary as removing outliers might not influence our model building. Therefore, we will move ahead without treating them.

Univariate analysis

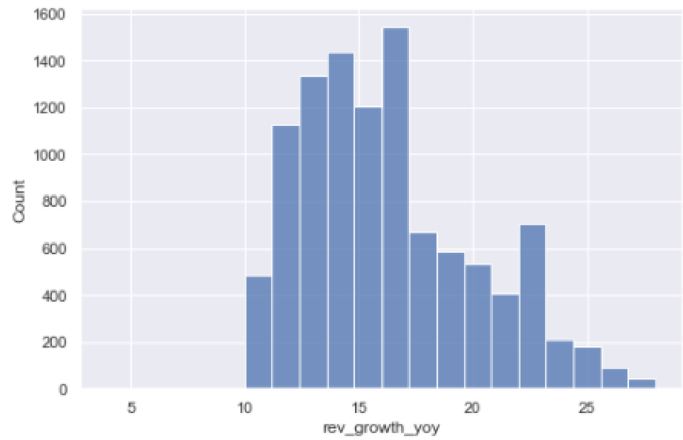
As, there are many variables present in the dataset, we will be checking analysis for some important variables only. The variables taken here for univariate analysis are based on the best VIF values.

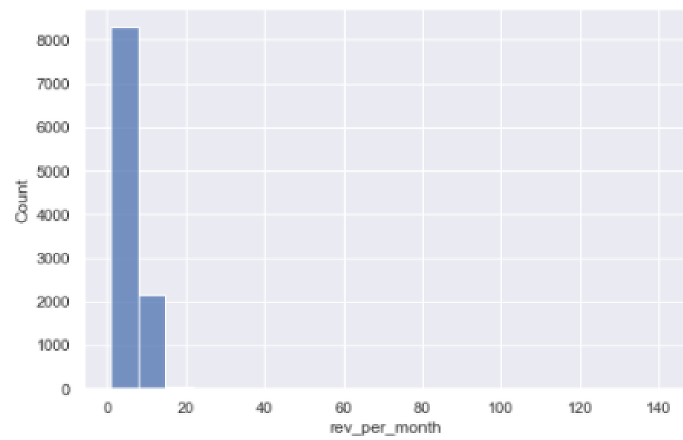
- **Tenure and Payment**



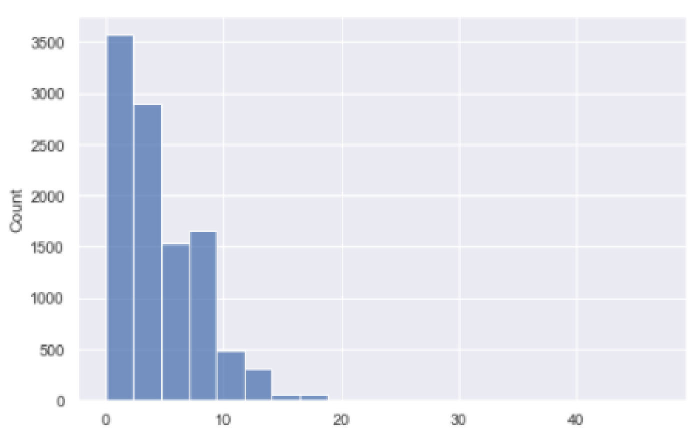


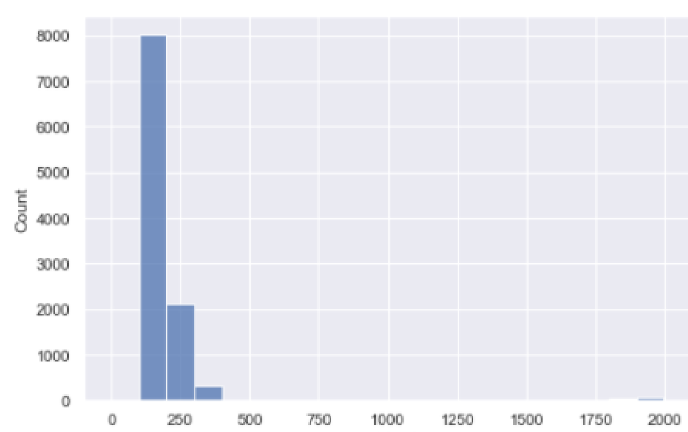
- **Rev_growth_yoy and rev_per_month**





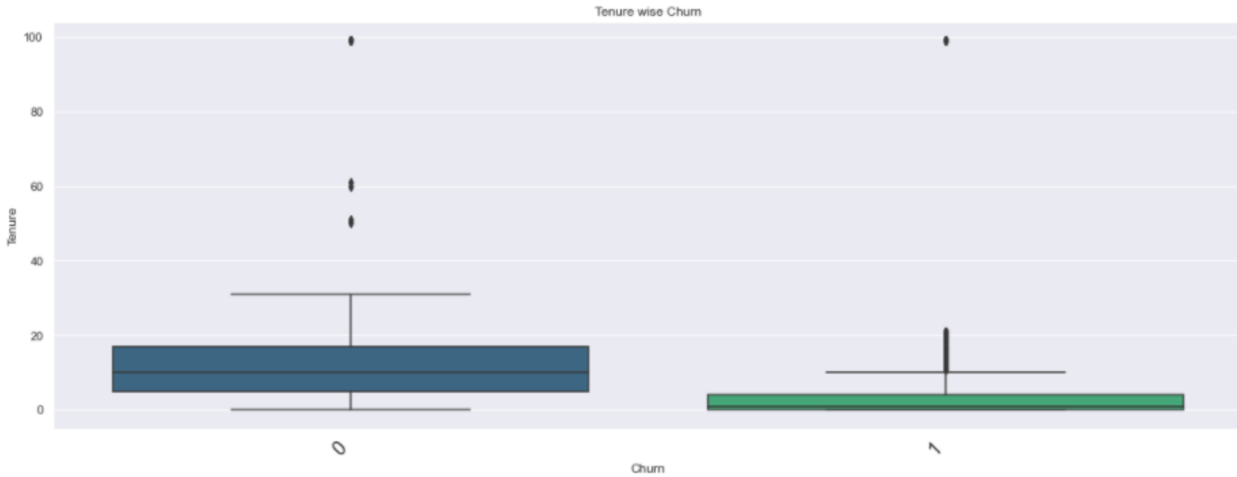
- **Days_since_CC_connect and cashback**



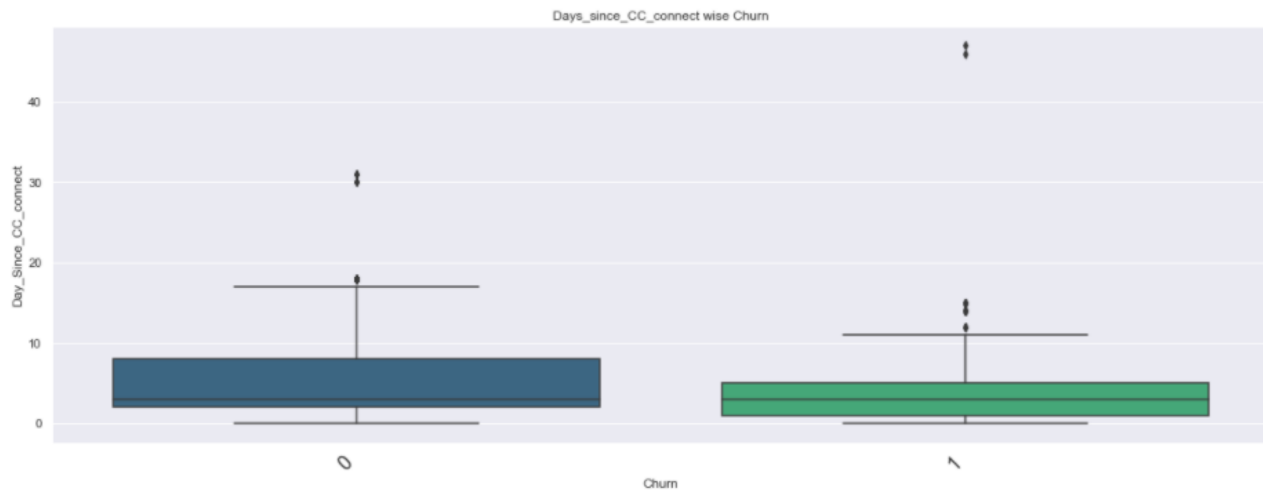


Bivariate and Multi variate Analysis-

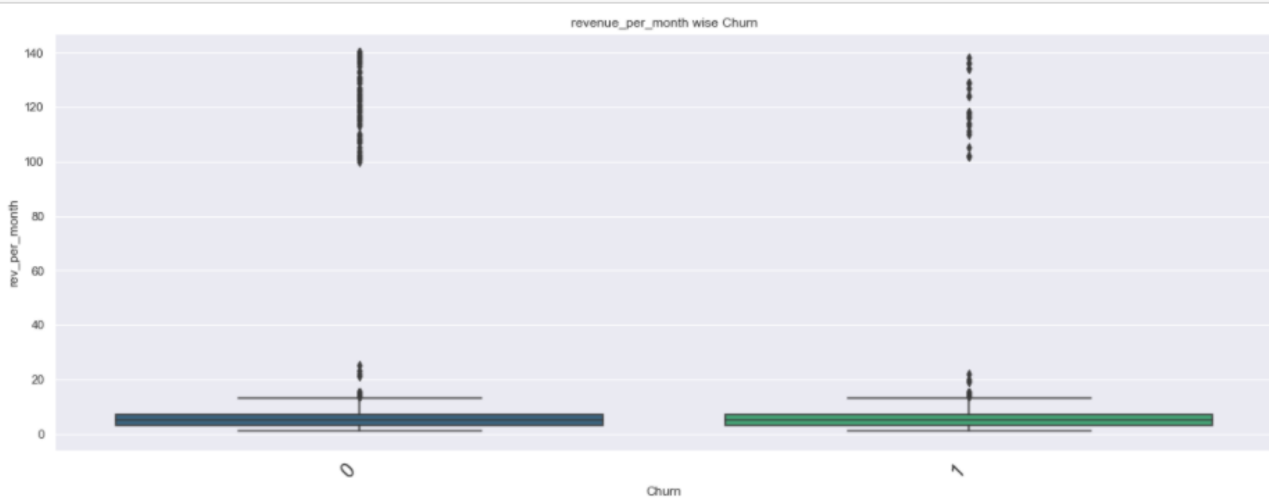
- **Churn with respect to Tenure**



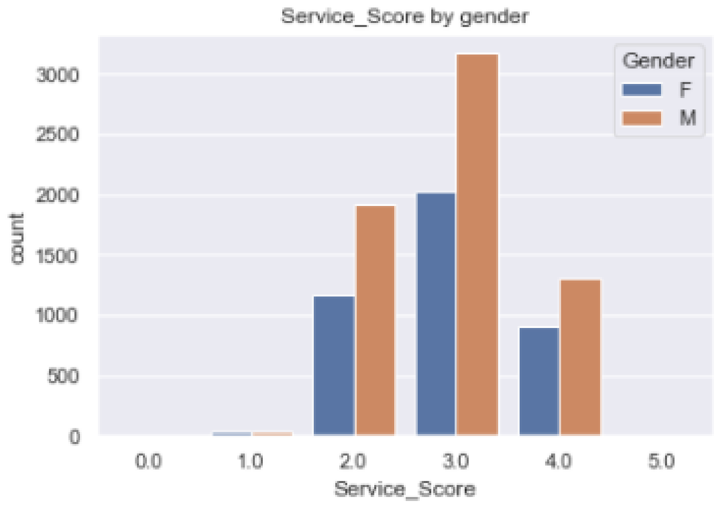
- **Churn with respect to Days_since_CC_connect**



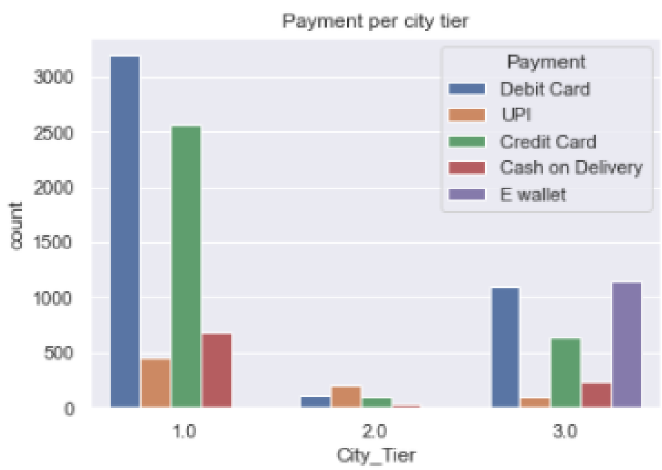
- **Churn with respect to rev_per_month**



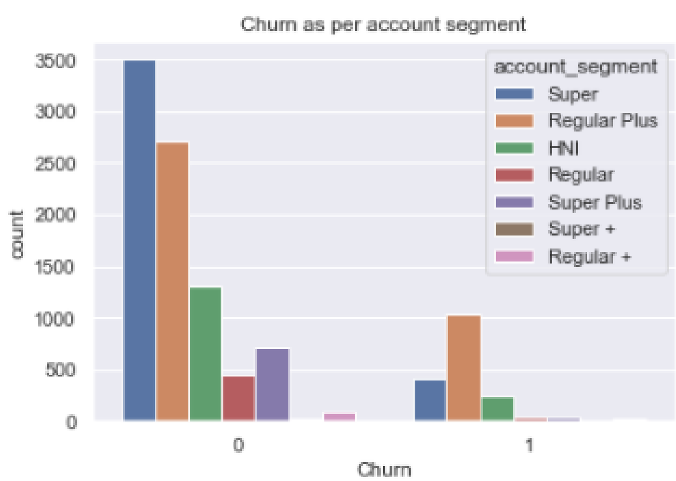
➢ Service Score by Gender



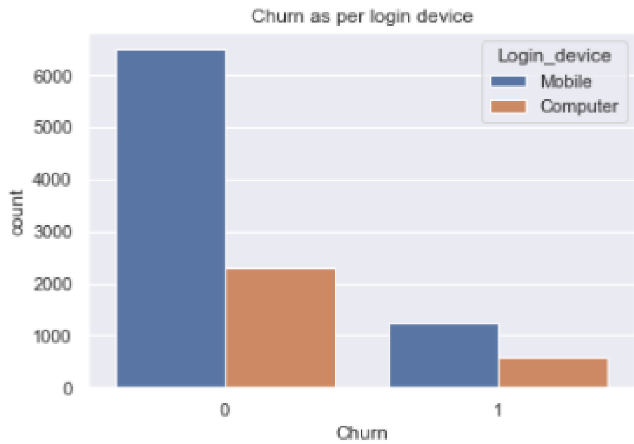
➢ Payment per City_Tier



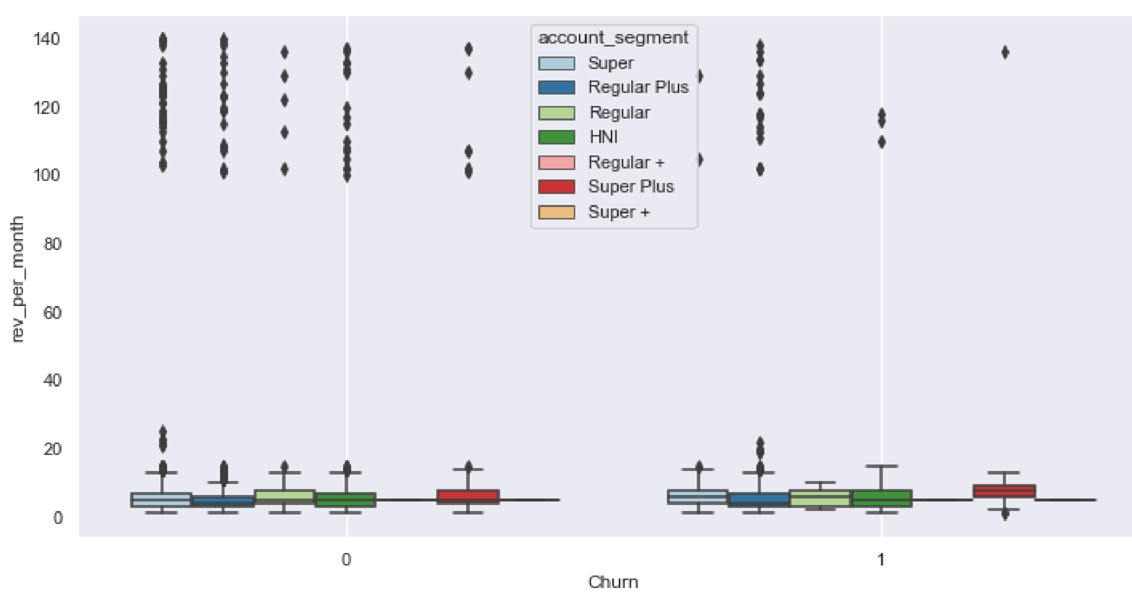
➢ Churn with respect to account_segment



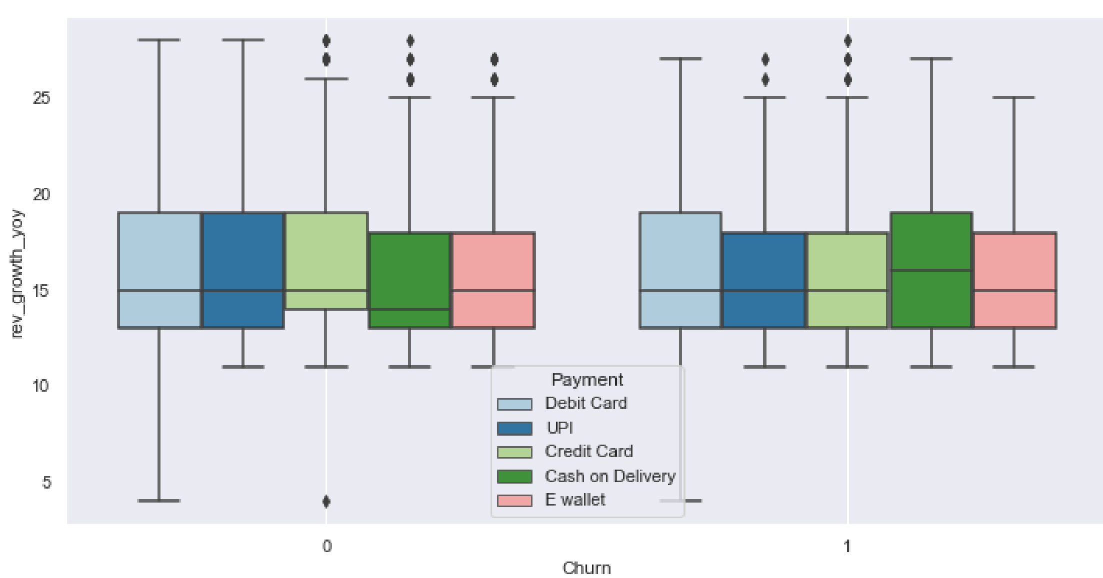
➢ Churn with respect to Login device



➢ Churn with respect to account segment and revenue per month



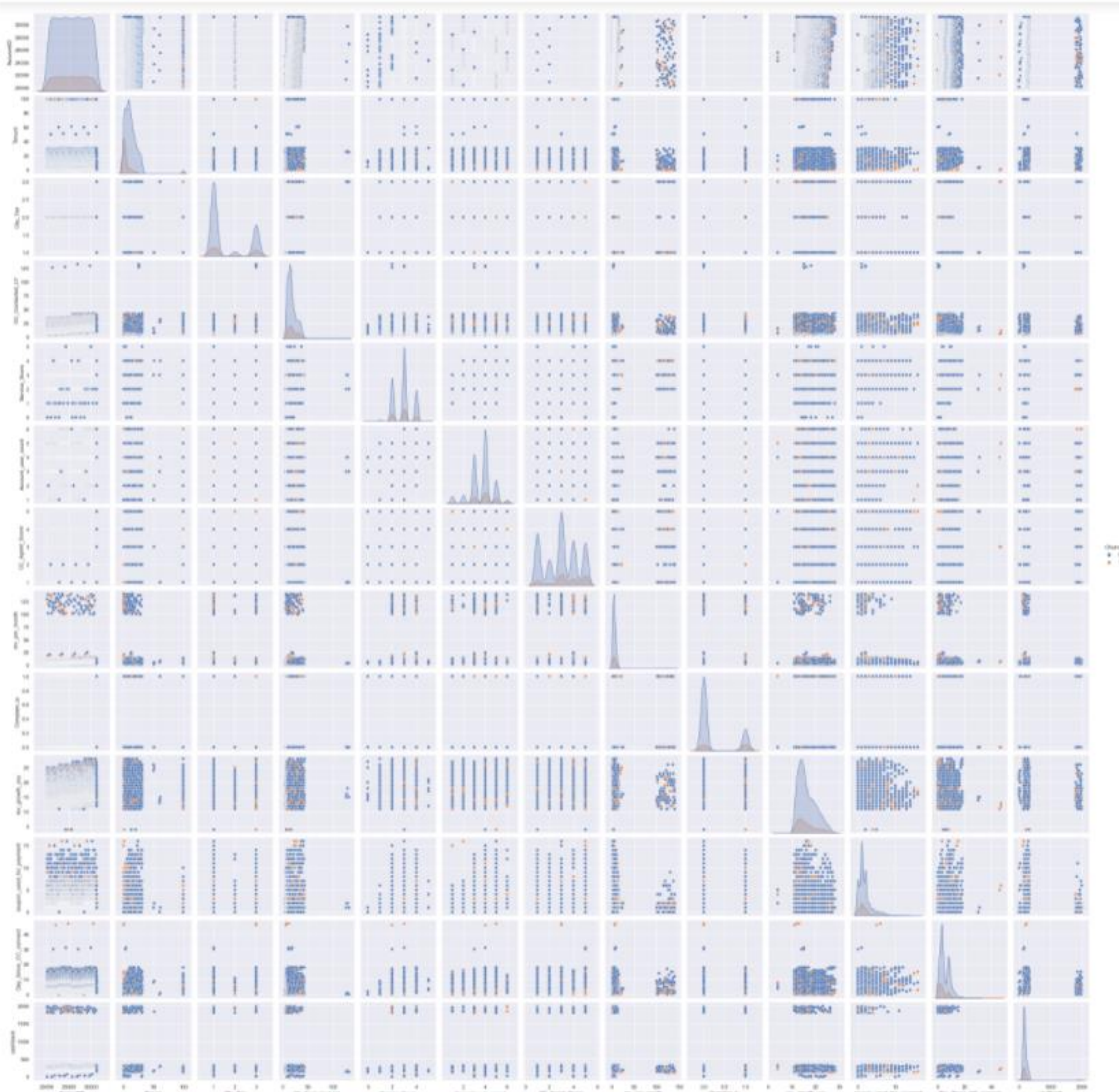
➢ Churn with respect to payment and revenue growth yoy



Pair-plot

We use pair plot to plot multiple pairwise bivariate distributions in a dataset. It creates a grid of Axes such that each numeric variable in data will be shared across the y-axes across a single row and the x-axes across a single column. The diagonal plots are treated differently: a univariate distribution plot is drawn to show the marginal distribution of the data in each column.

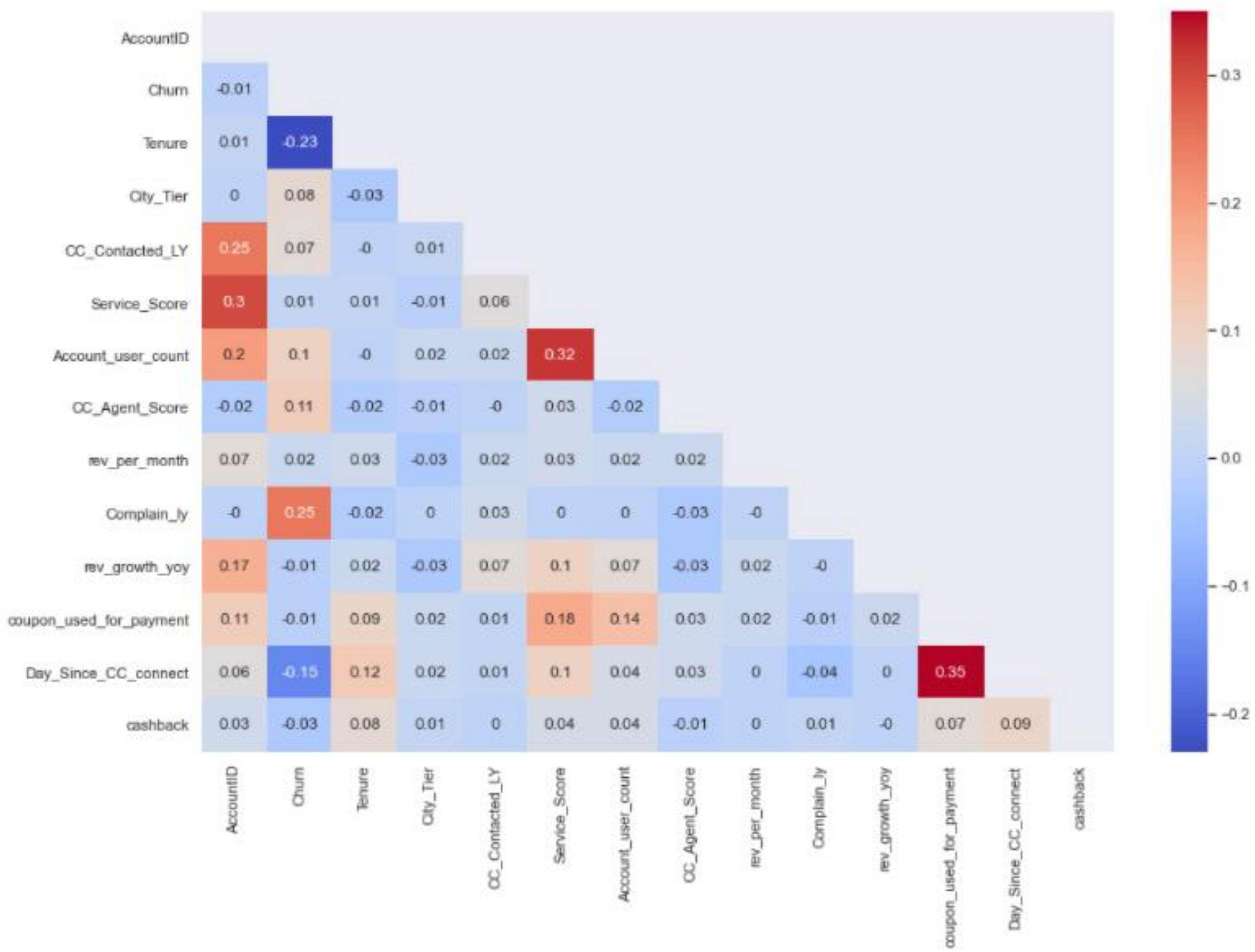
We are viewing the pair plot with Churn as a separator.



Correlation Matrix

	AccountID	Churn	Tenure	City_Tier	CC_Contacted_LY	Service_Score	Account_user_count	CC_Agent_Score	rev_per_month	Complain_ly	rev_growth_yoy
AccountID	1.00	-0.01	0.01	0.00	0.25	0.30	0.20	-0.02	0.07	-0.00	0.17
Churn	-0.01	1.00	-0.23	0.08	0.07	0.01	0.10	0.11	0.02	0.25	-0.01
Tenure	0.01	-0.23	1.00	-0.03	-0.00	0.01	-0.00	-0.02	0.03	-0.02	0.02
City_Tier	0.00	0.08	-0.03	1.00	0.01	-0.01	0.02	-0.01	-0.03	0.00	-0.03
C_Contacted_LY	0.25	0.07	-0.00	0.01	1.00	0.06	0.02	-0.00	0.02	0.03	0.07
Service_Score	0.30	0.01	0.01	-0.01	0.06	1.00	0.32	0.03	0.03	0.00	0.10
unt_user_count	0.20	0.10	-0.00	0.02	0.02	0.32	1.00	-0.02	0.02	0.00	0.07
C_Agent_Score	-0.02	0.11	-0.02	-0.01	-0.00	0.03	-0.02	1.00	0.02	-0.03	-0.03
rev_per_month	0.07	0.02	0.03	-0.03	0.02	0.03	0.02	0.02	1.00	-0.00	0.02
Complain_ly	-0.00	0.25	-0.02	0.00	0.03	0.00	0.00	-0.03	-0.00	1.00	-0.00
rev_growth_yoy	0.17	-0.01	0.02	-0.03	0.07	0.10	0.07	-0.03	0.02	-0.00	1.00

HeatMap



There is highly correlation between Tenure and Churn and Complain_ly and Churn.

```
Customer.Churn.value_counts()
```

```
0    9364
1    1896
Name: Churn, dtype: int64
```

Our dataset is imbalanced in terms of Churn response. It can be balanced using a technique called SMOTE.

Insights and recommendations based on basic exploration of data and analysis

- As per descriptive statistics, mean revenue per month is approx. 6 units whereas max is 140 units.
- On an average, users have received approx. 200 units as cashbacks.
- Mostly, the customers having lower tenures have churned.
- Maximum customers who have churned, holds a regular plus account.
- Payment type does not affect churn rate
- There seems to be very less correlation between variables as seen in the heat map.
- All variables seems to be a good differentiator for churn 0 and 1 as seen in the pair plot.
- Our data is highly imbalanced wherein we have 8794 customers who have not churned and only 1783 customers who have churned. In a way, it is good that the churn rate is less.
- We can populate this data using techniques and balance it but it won't represent the exact business problem.
- People with lower tenure have churned mostly for which I suggest the company can sell annual packages with good offers which will help have a lock in period for customers and hence lesser churn rate.
- Our data doesn't have more users from Tier 2 cities.
- Credit card is mostly used for payment in Tier 1 cities whereas Tier 3 cities mostly use e-wallet and debit card.
- Most customers who do not have churn have a super account whereas who churn have a Regular plus account.
- Complain_ly, Tenure and Days_since_CC_Connect have high correlation with Churn rate.

Now, we will get dummies of categorical columns and then convert all to float or integer for further model building.
The dataset info looks like below:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 11260 entries, 0 to 11259
Data columns (total 27 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   Account_user_count          11260 non-null  float64
 1   CC_Agent_Score              11260 non-null  float64
 2   CC_Contacted_LY             11260 non-null  float64
 3   Churn                       11260 non-null  int64
 4   City_Tier                   11260 non-null  float64
 5   Complain_ly                 11260 non-null  float64
 6   Day_Since_CC_connect        11260 non-null  float64
 7   Service_Score               11260 non-null  float64
 8   Tenure                      11260 non-null  float64
 9   cashback                    11260 non-null  float64
 10  coupon_used_for_payment     11260 non-null  float64
 11  rev_growth_yoy              11260 non-null  float64
 12  rev_per_month               11260 non-null  float64
 13  Payment_Credit Card         11260 non-null  uint8
 14  Payment_Debit Card          11260 non-null  uint8
 15  Payment_E wallet            11260 non-null  uint8
 16  Payment_UPI                 11260 non-null  uint8
 17  Gender_M                    11260 non-null  uint8
 18  account_segment_Regular     11260 non-null  uint8
 19  account_segment_Regular +   11260 non-null  uint8
 20  account_segment_Regular Plus 11260 non-null  uint8
 21  account_segment_Super       11260 non-null  uint8
 22  account_segment_Super +     11260 non-null  uint8
 23  account_segment_Super Plus  11260 non-null  uint8
 24  Marital_Status_Married      11260 non-null  uint8
 25  Marital_Status_Single       11260 non-null  uint8
 26  Login_device_Mobile         11260 non-null  uint8
dtypes: float64(12), int64(1), uint8(14)
memory usage: 1.4 MB
```

After getting our data ready for model building, we can check for clusters if any by using K Means clustering algorithm.

Clus_kmeans2	Account_user_count	CC_Agent_Score	CC_Contacted_LY	Churn	City_Tier	Complain_ly	Day_Since_CC_connect	Service_Score	Tenure
0	3.704986	3.067611	17.849713	0.168042	1.647866	0.275825	4.583931	2.903246	10.992557
1	3.703704	2.879630	17.898148	0.203704	1.601852	0.324074	4.305556	2.916667	10.296296

ent_Regular Plus	account_segment_Super	account_segment_Super +	account_segment_Super Plus	Marital_Status_Married	Marital_Status_Single	Login_device_Mobile	freq
0.342719	0.369440	0.004214	0.068060	0.539186	0.312859	0.731169	11152
0.370370	0.361111	0.000000	0.111111	0.546296	0.287037	0.814815	108

The last column i.e. freq shows that there are 11152 records in one cluster and only 108 in another cluster. This arises because our data is highly imbalanced.
Also, the values of all attributes shows no visible difference between both clusters. Therefore, we cannot devise any conclusions through this clustering.

Data Splitting

The train-test split is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm.

The procedure involves taking a dataset and dividing it into two subsets. The first subset is used to fit the model and is referred to as the **training dataset**. The second subset is not used to train the model; instead, the input element of the dataset is provided to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the **test dataset.**

Train Dataset: Used to fit the machine learning model.

Test Dataset: Used to evaluate the fit machine learning model.

The objective is to estimate the performance of the machine learning model on new data: data not used to train the model.

Here, we have Churn as our target variable i.e. which is to be predicted.

Therefore, we will separate this column from our dataset before splitting it into train and test data.

```
from sklearn.model_selection import train_test_split
# Split the data into training and test set in 70:30 ratio
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30 , random_state=1,stratify=y)
```

```
y_train.value_counts(1)
```

```
0    0.831642
1    0.168358
Name: Churn, dtype: float64
```

Our target variable has a ratio of 84% and 16%.

We need to balance our data before building the model to get a better analysis. We will use a technique called Synthetic Minority Oversampling Technique, or SMOTE.

The simplest approach involves duplicating examples in the minority class, although these examples don't add any new information to the model. Instead, new examples can be synthesized from the existing examples.

SMOTE works by selecting examples that are close in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line.

A general downside of the approach is that synthetic examples are created without considering the majority class, possibly resulting in ambiguous examples if there is a strong overlap for the classes.

```
# After balancing the data Here is how the Balanced percentage of Data Looks Like
(y_train_res.value_counts()/len(y_train_res.index))*100
```

```
1    50.0
0    50.0
Name: Churn, dtype: float64
```

Thus, we can say that we have a balanced data.

Now, on the new balanced splitted training and testing data we will be used in building different predictive models like Random Forest, Logistic Regression, Ada Boost and XG Boost model.

1. Random Forest Classifier

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. We are using below parameters to build this model.

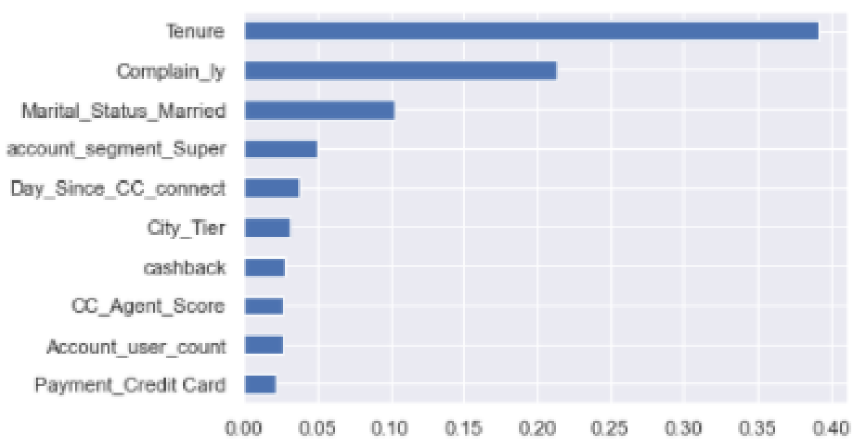
```
seed = 0
# Random Forest parameters
rf_params = {
    'n_jobs': -1,
    'n_estimators': 1000,
    'max_features': 0.3,
    'max_depth': 4,
    'min_samples_leaf': 2,
    'max_features' : 'sqrt',
    'random_state' : seed,
    'verbose': 1
}
```

Performance Metrics:

```
Accuracy score: 0.8658969804618117
=================================================================
              precision    recall  f1-score   support

           0       0.95      0.89      0.92      2809
           1       0.58      0.75      0.65       569

    accuracy                           0.87      3378
   macro avg       0.76      0.82      0.78      3378
weighted avg       0.88      0.87      0.87      3378
```

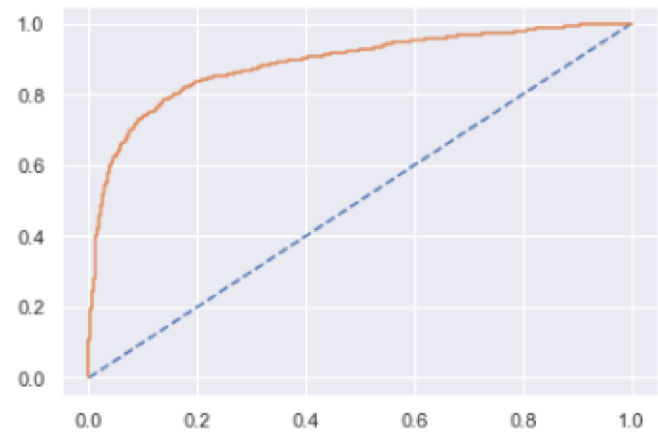
Here, the Recall score is good in predicting both 0 and 1. But, more support is towards predicting customers who will not churn which is not of much importance.

Feature Importance:



This shows that Tenure and Complain_ly are the most important features in predicting the Churn rate.

```
Area under Curve is 0.8877384455312793
```



2. <u>Logistic Regression Model</u>

Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. The most common logistic regression models a binary outcome; something that can take two values such as true/false, yes/no, and so on.

```
LRmodel = LogisticRegression()
LRmodel.fit(X_train_res, y_train_res)
```

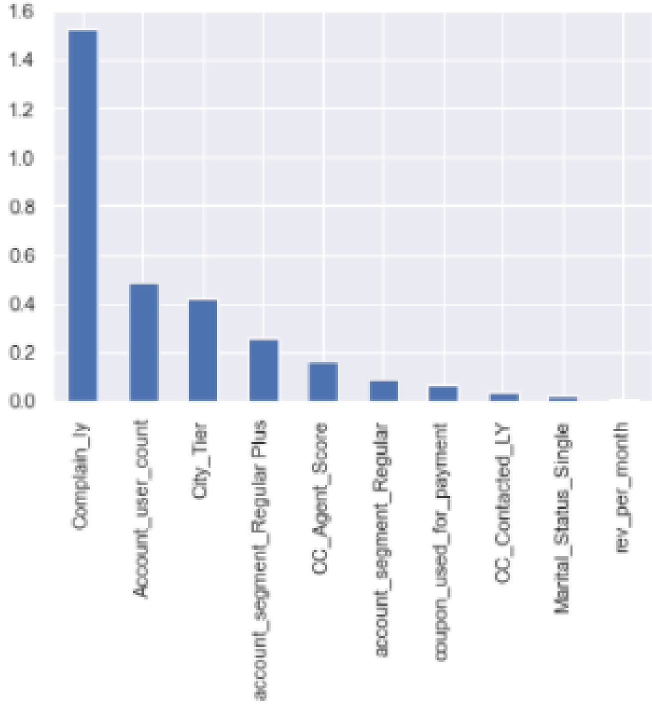
```
LogisticRegression()
```

```
ytrain_predictLogit = LRmodel.predict(X_train_res)
ytest_predictLogit = LRmodel.predict(X_test)
```

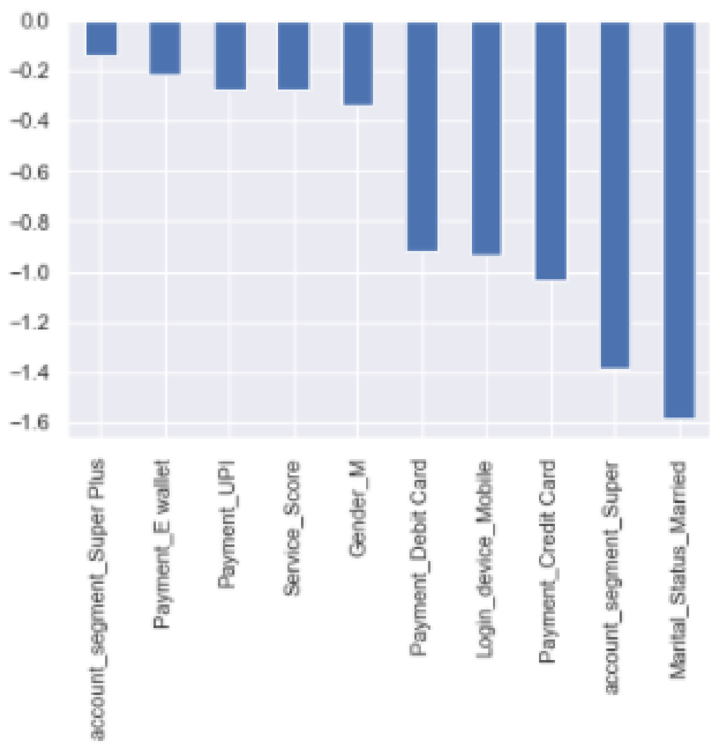
Performance Metrics:

```
Accuracy score: 0.7904085257548845
================================================================================
              precision    recall  f1-score   support

           0       0.93      0.81      0.87      2809
           1       0.42      0.69      0.53       569

    accuracy                           0.79      3378
   macro avg       0.68      0.75      0.70      3378
weighted avg       0.84      0.79      0.81      3378
```

Here, the recall and accuracy is not as good as it was in previous model of random forest.



Also, it shows a difference in important features. According to Logistic regression, Tenure is not the most importance feature, but Complain_ly is.



There, is also a negative interdependency of married, marital status on Churn rate.

Model Tuning

We will apply grid search and find out the best parameters for building the model on given data.
The best parameters comes out to be as below:

```
print(grid_search.best_params_,'\n')
print(grid_search.best_estimator_)
```

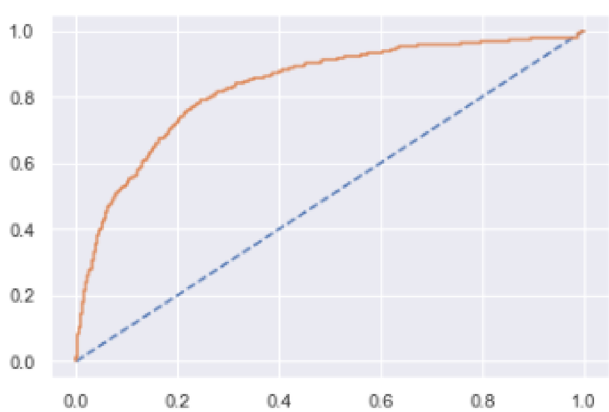
```
{'penalty': 'l2', 'solver': 'liblinear', 'tol': 0.0001}

LogisticRegression(max_iter=10000, n_jobs=2, solver='liblinear')
```

```
Accuracy score: 0.8161634103019538
========================================================================
              precision    recall  f1-score   support

           0       0.92      0.85      0.89      2809
           1       0.47      0.63      0.54       569

    accuracy                           0.82      3378
   macro avg       0.69      0.74      0.71      3378
weighted avg       0.84      0.82      0.83      3378
```

The accuracy has increased a bit but still lesser than that of Random Forest Model.

Also, other metrics like recall, precision are also lower.

```
Area under Curve is 0.8320706541426909
```



3. Ada Boost Model

Boosting is an ensemble technique that attempts to create a strong classifier from a number of weak classifiers. In Ada Boost model, weak models are added sequentially, trained using the weighted training data.

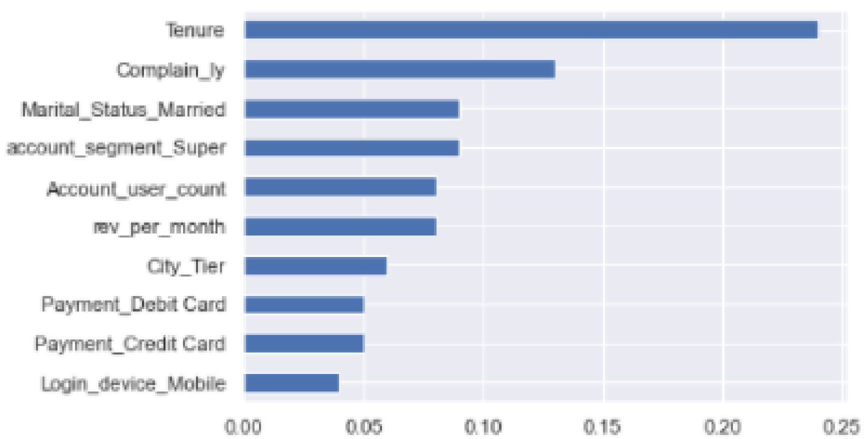
The process continues until a pre-set number of weak learners have been created (a user parameter) or no further improvement can be made on the training dataset.

Once completed, you are left with a pool of weak learners each with a stage value.

Performance Metrics:

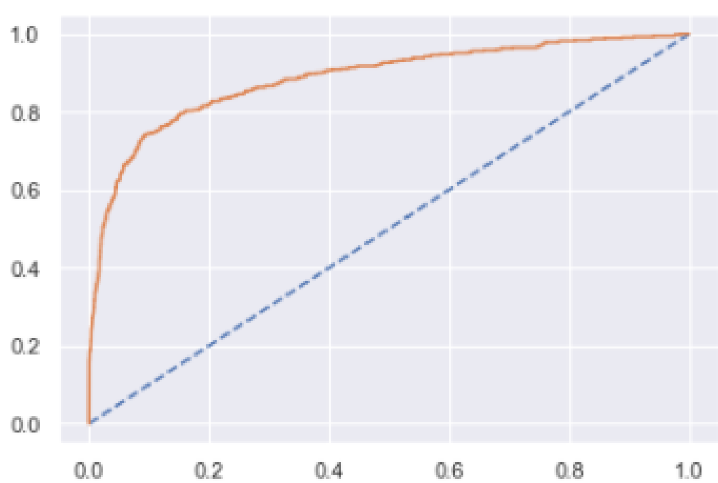
```
Accuracy score: 0.8552397868561279
==============================================================
              precision    recall  f1-score   support

           0       0.95      0.87      0.91      2809
           1       0.55      0.76      0.64       569

    accuracy                           0.86      3378
   macro avg       0.75      0.82      0.77      3378
weighted avg       0.88      0.86      0.86      3378
```

Here, the recall and accuracy is much lesser than it was in previous models



Tenure and Complain_ly are the most important features.

```
Area under Curve is 0.8856568861949508
```



4. XG Boost Model

XG Boost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework

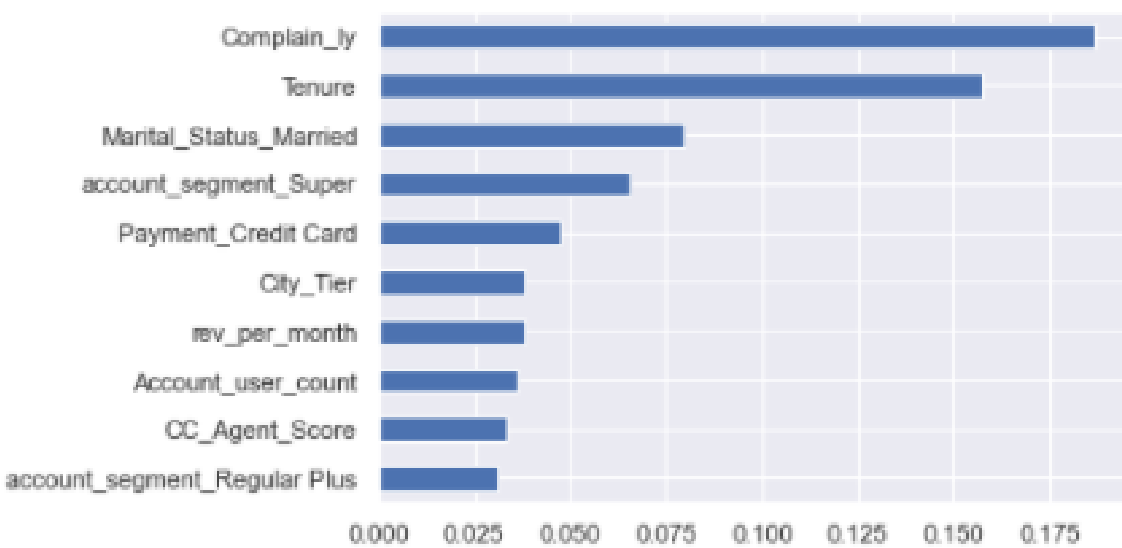
```
import xgboost
classifier=xgboost.XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=0.5, gamma=0.4, learning_rate=0.1,
       max_delta_step=0, max_depth=6, min_child_weight=7,
       n_estimators=100, n_jobs=1,
       objective='binary:logistic')
```

Performance Metrics:

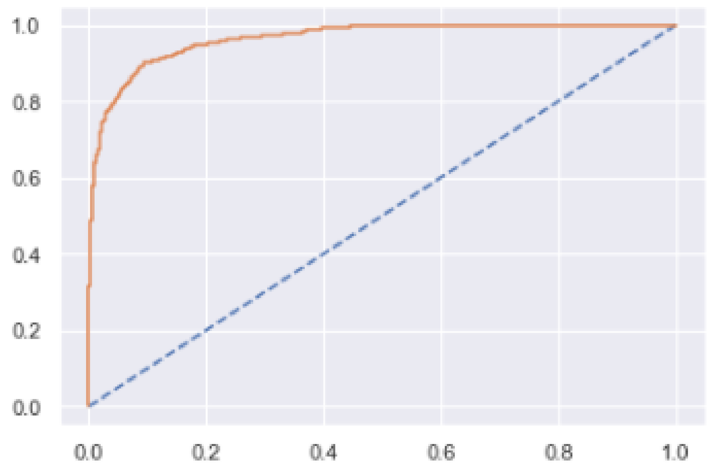
```
Accuracy score: 0.9378330373001776
==============================================================
              precision    recall  f1-score   support

           0       0.95      0.97      0.96      2809
           1       0.85      0.77      0.81       569

    accuracy                           0.94      3378
   macro avg       0.90      0.87      0.88      3378
weighted avg       0.94      0.94      0.94      3378
```

This model has the highest accuracy and other metrics amongst all.

Hence, we will go ahead with this one for our churn prediction.



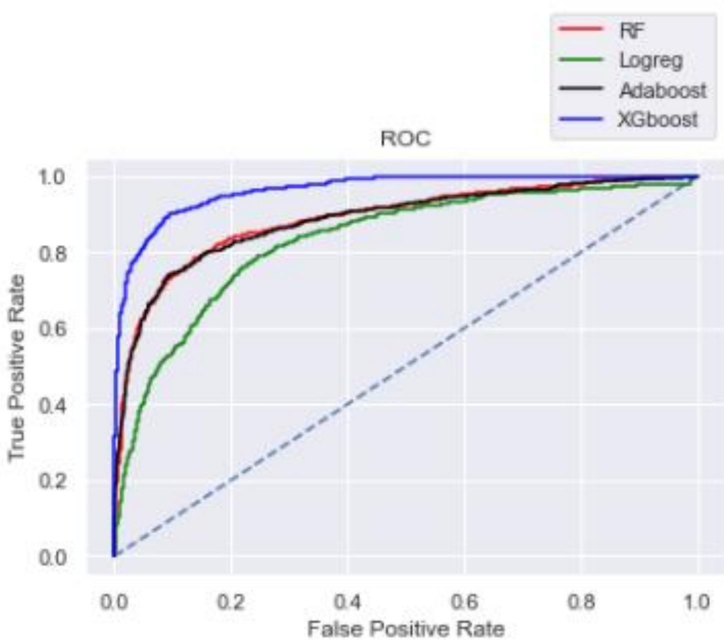
```
Area under Curve is 0.9644358048226858
```



Comparison of all the models

	Random Forest	Logit Test	Ada Boost test	XG Boost Test
Accuracy	0.87	0.82	0.86	0.94
Recall	0.75	0.63	0.76	0.77
Precision	0.58	0.47	0.55	0.85
F1 Score	0.65	0.54	0.64	0.81
AUC	0.89	0.83	0.89	0.96

XG Boost model is the winner and shows a very good accuracy score.



Does my Model Predict?????

For checking this, we are taking a small set of observations as our sample data.

```
m1 =custdataset_final[2:7]
m1
```

	Account_user_count	CC_Agent_Score	CC_Contacted_LY	Churn	City_Tier	Complain_ly	Day_Since_CC_connect	Service_Score	Tenure	cashback	...	accou
2	4.0	3.0	30.0	1	1.0	1.0	3.0	2.0	0.0	165.25	...	
3	4.0	5.0	15.0	1	3.0	0.0	3.0	2.0	0.0	134.07	...	
4	3.0	5.0	12.0	1	1.0	0.0	3.0	2.0	0.0	129.60	...	
5	4.0	5.0	22.0	1	1.0	1.0	7.0	3.0	0.0	139.19	...	
6	3.0	2.0	11.0	1	3.0	0.0	0.0	2.0	2.0	120.86	...	

5 rows × 28 columns

We have split it into test and train data

```
m2 =m1.drop('Churn', axis=1)
m3 = m1['Churn']
```

And then, fit it into XG boost classifier to predict the Churn for test data.

```
classifier.fit(m2,m3)
m4= classifier.predict(m2)
print(m4)
```

```
[21:36:00] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/src/learner.cc:1095: Starting in XGBoost 1.3.
0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly se
t eval_metric if you'd like to restore the old behavior.
[1 1 1 1 1]
```

As we can see that our model has predicted all the 1s for Churn correctly.

Business Recommendations

- We can use XG Boost model for our prediction.
- Tenure and Complain_ly are the most important features.
- The company should focus on giving discount or offers plans with long tenures because if a customer stays with a company for long they tend to become loyal users.
- Company should also focus on working on the customer service. If a customer has complaint in the past year then they are most likely to churn.
- If customer service is strong then their complaints can be resolved and the customer will stay.
- Churn rate is also somehow related to marital status.This needs to be observed and checked by doing some customer survey and can find out the reason.
- Also, people who have account segment as Super mostly do not churn. It can be because of many reasons like price, value it brings to the customer, customisation, etc. The company can make more such plans and promote to the customers of various demographics.
- The customers who churn more have a Regular plus account.

--THE END--