Credit Card Customers Analysis

By Aryan, Vaibhav and Harsh

Business Problem Statement

A bank manager is uncomfortable with more and more customers leaving their credit card services. They would really appreciate it if someone could predict who will be affected so that they can proactively go to the customer to provide them with better services and turn customer decisions in the opposite direction.

Goal

This project involves the analysis of Credit card customer dataset which contains the information about 10127 Customer transaction.

With this dataset, we have to make a Classification ML model to predict the Attrition customer.

Data description

1. CLIENTNUM: Client number. Unique identifier for the customer holding the account
2. Customer_Age: Demographic variable - Customer's Age in Years
3. Gender: Demographic variable - M=Male, F=Female
4. Dependent_count: Demographic variable - Number of dependents
5. Education_Level: Demographic variable - Educational Qualification of the account holder (example: high school, college graduate, etc.)
6. Marital_Status: Demographic variable - Married, Single, Divorced, Unknown
7. Income_Category: Demographic variable - Annual Income Category of the account holder
8. Card_Category: Product Variable - Type of Card (Blue, Silver, Gold, Platinum)
9. Months_on_book: Period of relationship with bank
10. Total_Relationship_Count: Total no. of products held by the customer
11. Months_Inactive_12_mon: No. of months inactive in the last 12 months
12. Contacts_Count_12_mon: No. of Contacts in the last 12 months
13. Credit_Limit: Credit Limit on the Credit Card
14. Total_Revolving_Bal: Total Revolving Balance on the Credit Card
15. Avg_Open_To_Buy: Open to Buy Credit Line (Average of last 12 months)
16. Total_Amt_Chng_Q4_Q1: Change in Transaction Amount (Q4 over Q1)
17. Total_Trans_Amt: Total Transaction Amount (Last 12 months)
18. Total_Trans_Ct: Total Transaction Count (Last 12 months)
19. Total_Ct_Chng_Q4_Q1: Change in Transaction Count (Q4 over Q1)
20. Avg_Utilization_Ratio: Average Card Utilization Ratio

IMPORTING LIBRARIES

Importing Data

we will import our data and exclude our last 2 features as they are feature of some different model and **CLIENTNUM** feature (As it is a unique identifier in our dataset).

CLEANING and PREPROCESSING

```
Out[5]: Attrition_Flag              0
        Customer_Age                0
        Gender                      0
        Dependent_count             0
        Education_Level             0
        Marital_Status              0
        Income_Category             0
        Card_Category               0
        Months_on_book              0
        Total_Relationship_Count    0
        Months_Inactive_12_mon      0
        Contacts_Count_12_mon       0
        Credit_Limit                0
        Total_Revolving_Bal         0
        Avg_Open_To_Buy             0
        Total_Amt_Chng_Q4_Q1        0
        Total_Trans_Amt             0
        Total_Trans_Ct              0
        Total_Ct_Chng_Q4_Q1         0
        Avg_Utilization_Ratio       0
        dtype: int64
```

After doing some cleaning of our dataset we found out that we don't have any missing value, Null value and no duplicate rows in our dataset.

EXPLORATORY DATA ANALYSIS

Before EDA we first divide our dataset into Numerical (In Numerical we sub-divide into Discrete and Continuous feature) and Categorial feature.

As we can do group wise analysis of our dataset.

```
Number of Numerical feature: 14

Number of Discrete feature: 4
```

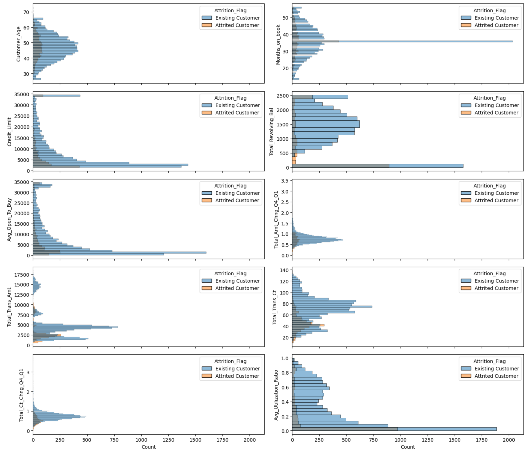
```
Out[9]: Dependent_count             6
        Total_Relationship_Count    6
        Months_Inactive_12_mon      7
        Contacts_Count_12_mon       7
        dtype: int64
```

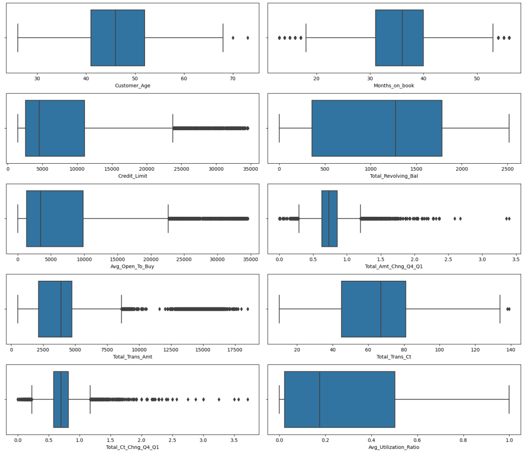
```
Number of Continuous feature: 10

Number of categorical feature: 6
```

So,we have 14 Numerical Feature and 6 Categorical feature. Now, let's start our analysis

ANALYSIS OF CONTINOUS AND DISCRETE FEATURES

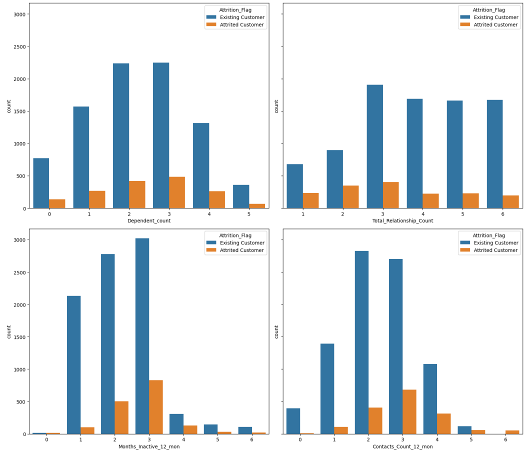
For this we will plot histogram to see the distribution of of our continuous feature.



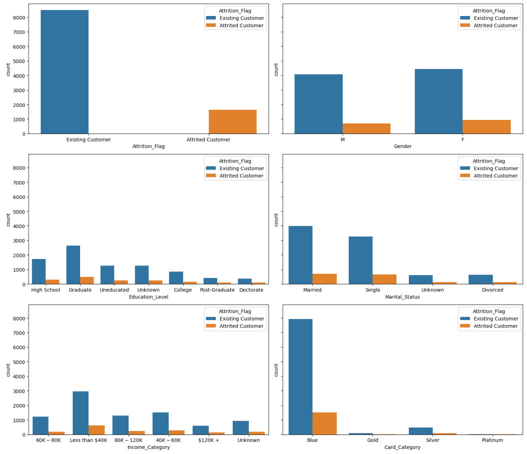


As our Continuous feature are very skewed by the distribution of histogram and with the help of Boxplot we also able to see outliers and skewedness but we won't be removing as they are making up big Chunk in our dataset.

Now let's take a look on discrete feature



ANALYSIS OF CATEGORICAL FEATURES



Obervation we get from this:

1. As People are getting more educated, they are less likely to use credit card and their attrition count are also getting decreased.
2. People having income less than 40k have significantly higher attrition count.
3. People having Blue Credit are also having significantly higher attrition count.
4. Also our dataset is also imbalanced as Existing customer are much higher than Attrited customer, we will deal with this problem later in our study.

STATISTICAL ANALYSIS

Now let's see how each features affect one another in our dataset.

```
    For this we will apply some statistical test:
    1. For Analysing categorical to categorical
feature relationship, we will use Chi-squared test.
    2. For Analysis categorical to Numerical
feature relationship, we will use One way anova.
```

Chi squared test of independence

For Gender and Income category P<0.05, then we reject our null hypothesis that Gender and attrition flag are independent. SO, Gender and Attrition are depend on each other.

For Education level, marital Status and card category P>0.05, then we accept our null hypothesis that Card_Category and attrition flag are independent

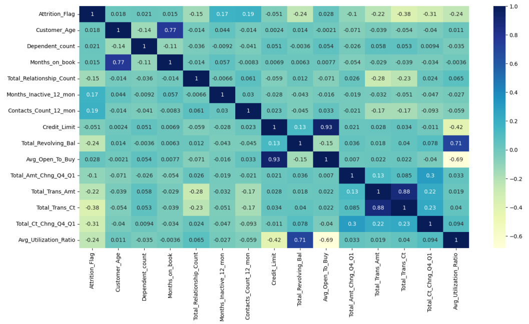
Anova Test

Since p>0.05, we can say that Attrition chance is not statistically associated with customer Age, Months_Inactive_12_mon, Dependent_count, Avg_Open_To_Buy.

For other feature which has p < 0.05 we state that we have a main interaction effect. This simply means that amongst the groups at least any of the group(or groups) means statistically significantly differ from one another

HEATMAP

Now we will see features are correlated to one another.



CONCLUSION FROM HEATMAP

In our dataset, We applied spearman correleation and these incites we have found:

1. Avg_Open_To_Buy and Credit limit has the most correlation (93%)
2. Followed by, Total_Ct_chng_Q4_Q1 and Total_Trans_Amt (88%) and Month_On_Book and Customer_Age (77%)
3. Total_Revolving_Bal and Avg_Open_To_Buy has close to 70% correlation with Avg_Utilization_Ratio
4. Compare to these all other correlation are not that signification

Feature ENGINEERING

Now let's convert our Categorical features in numerical features.

In this case we will use one hot encoding

SPLITTING ARE DATASET IN TRAINING AND TESTING DATA

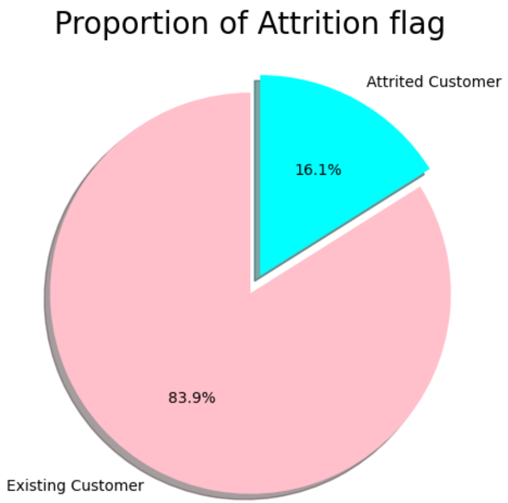
We will divide are dataset in 20% for testing and 80% for training.

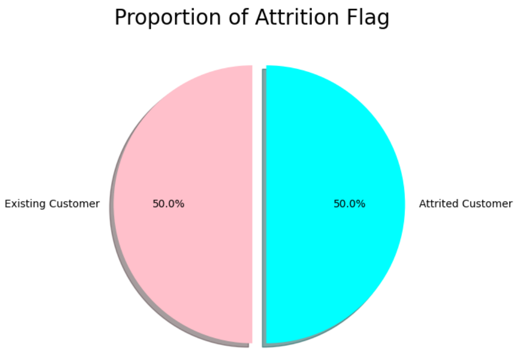
FEATURE SCALING

Now, we will scale our features so that every will be on same footing without giving importance to other features

SMOTE

As our dataset is imbalanced to fix this issue we will use oversampling to get 50-50% Attrited and Existing Customer.





We have store Smote features in Normalised dataset and Without Normalised dataset as some ML algorithm work better without Normalising.

Now our Dataset is ready for applying Modelling.

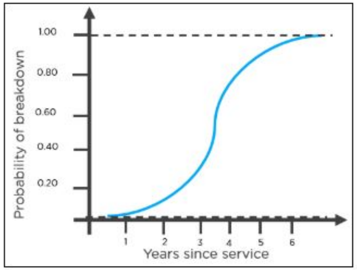
MODELLING

For Modelling our Dataset, we will apply 4 ML Model(Logistic regression, Decision Tree,KNN, Random Forest) and evaluating which one is performing well.

LOGISTIC REGRESSION

Logistic regression is a statistical method that is used for building machine learning models where the dependent variable is dichotomous: i.e. binary. Logistic regression is used to describe data and the relationship between one dependent variable and one or more independent variables. The independent variables can be nominal, ordinal, or of interval type.

The name “logistic regression” is derived from the concept of the logistic function that it uses. The logistic function is also known as the sigmoid function. The value of this logistic function lies between zero and one.



HYPER PARAMETER OF LOGISTIC REGRESSION

Logistic regression does not really have any critical hyperparameters to tune.

Sometimes, you can see useful differences in performance or convergence with different solvers (solver).

```
solver in [‘newton-cg’, ‘lbfgs’, ‘liblinear’,
‘sag’, ‘saga’]
```

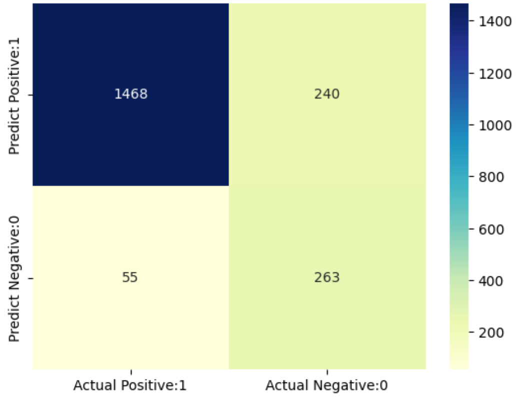
Regularization (penalty) can sometimes be helpful.

```
penalty in [‘none’, ‘l1’, ‘l2’, ‘elasticnet’]
```

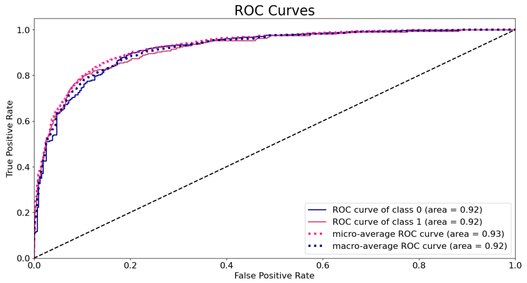
The C parameter controls the penality strength, which can also be effective.

```
C in [100, 10, 1.0, 0.1, 0.01]
```

Now,We have got best parameter we will use them to make our model.



```
              precision    recall  f1-score   support

           0       0.96      0.86      0.91      1708
           1       0.52      0.83      0.64       318

    accuracy                           0.85      2026
   macro avg       0.74      0.84      0.77      2026
weighted avg       0.89      0.85      0.87      2026
```

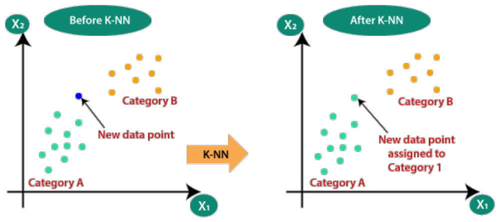


After applying Logistic regression model we are getting accuracy 85%

KNN

The k-nearest neighbors algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

For classification problems, a class label is assigned on the basis of a majority vote—i.e. the label that is most frequently represented around a given data point is used. While this is technically considered “plurality voting”, the term, “majority vote” is more commonly used in literature. The distinction between these terminologies is that “majority voting” technically requires a majority of greater than 50%, which primarily works when there are only two categories.



HYPER PARAMETER OF KNN

The most important hyperparameter for KNN is the number of neighbors (n_neighbors).

Test values between at least 1 and 21, perhaps just the odd numbers.

```
n_neighbors in [1 to 21]
```

It may also be interesting to test different distance metrics (metric) for choosing the composition of the neighborhood.

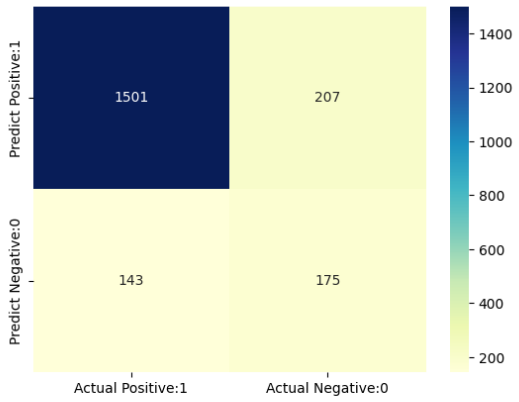
```
metric in [‘euclidean’, ‘manhattan’, ‘minkowski’]
```

It may also be interesting to test the contribution of members of the neighborhood via different weightings (weights).

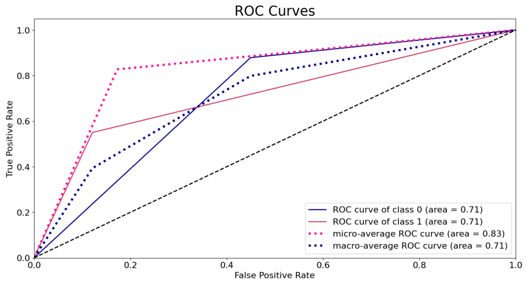
```
weights in [‘uniform’, ‘distance’]
```

```
Out[39]: ▾                  KNeighborsClassifier
         KNeighborsClassifier(metric='manhattan', n_neighbors=1)
```

```
Out[42]: <AxesSubplot: >
```



```
              precision    recall  f1-score   support

           0       0.91      0.88      0.90      1708
           1       0.46      0.55      0.50       318

    accuracy                           0.83      2026
   macro avg       0.69      0.71      0.70      2026
weighted avg       0.84      0.83      0.83      2026
```



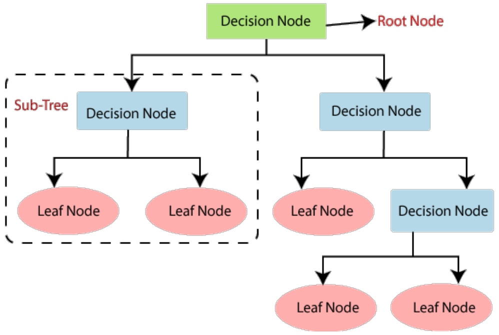
These are the result that we are getting for KNN model

DECISION TREE

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too.

The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).

In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.



HYPER PARAMETER OF DECISION TREES

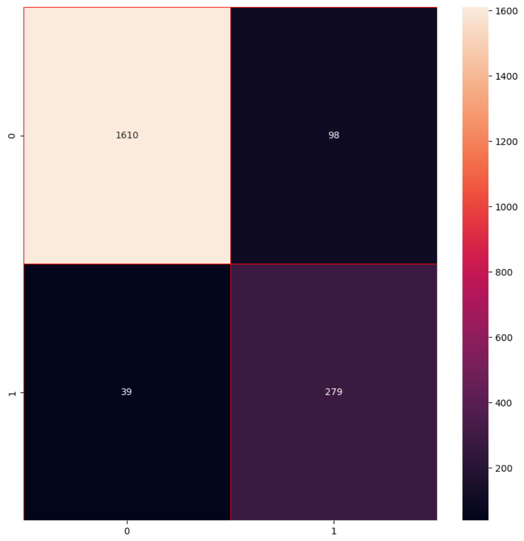
This argument represents the maximum depth of a tree. If not specified, the tree is expanded until the last leaf nodes contain a single value. Hence by reducing this meter, we can preclude the tree from learning all training samples thereby, preventing over-fitting.

Minimum samples split decides or hold the value for the minimum number of samples necessary to split a nonterminal node.4

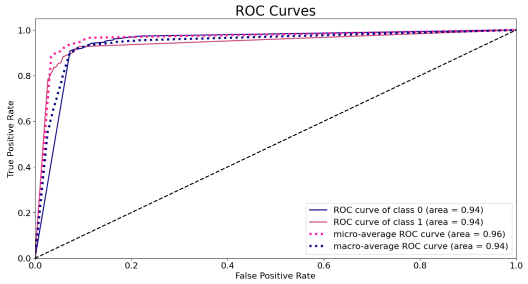
Minimum sample leaf may sound like minimum sample split and is somewhat similar too. But in this case, we are talking about the minimum number of samples required to be left at the leaf node.

```
Out[47]: ▾                DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=15, m
in_samples_leaf=5,
                       min_samples_split=16)
```

```
Confusion matrix

 [[1610   98]
 [  39  279]]
```



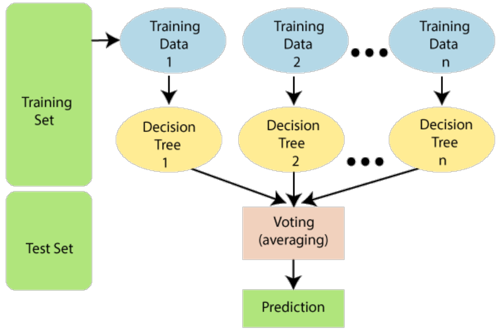
```
              precision    recall  f1-score   support

           0       0.98      0.94      0.96      1708
           1       0.74      0.88      0.80       318

    accuracy                           0.93      2026
   macro avg       0.86      0.91      0.88      2026
weighted avg       0.94      0.93      0.93      2026
```



For decision Tree model we are getting accuracy of 93%

Random FOREST

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction



HYPER PARAMETER OF RANDOM FOREST

The most important parameter is the number of random features to sample at each split point (max_features).

You could try a range of integer values, such as 1 to 20, or 1 to half the number of input features.

```
max_features [1 to 20]
```

Alternately, you could try a suite of different default value calculators.

```
max_features in ['sqrt', 'log2']
```

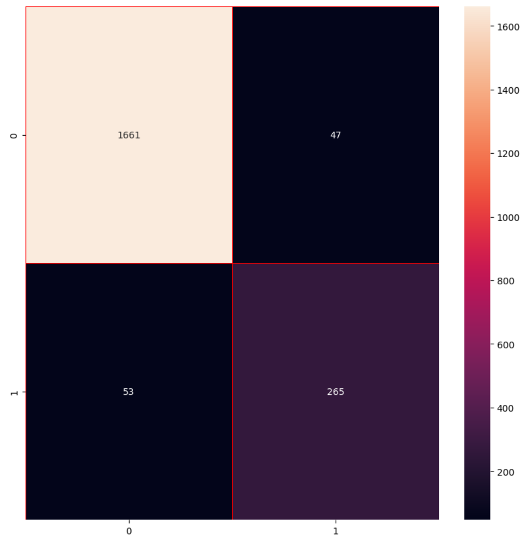
Another important parameter for random forest is the number of trees (n_estimators).

Ideally, this should be increased until no further improvement is seen in the model.

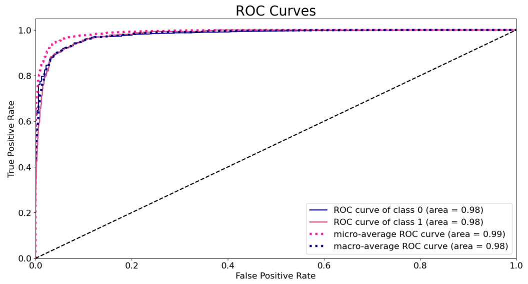
Good values might be a log scale from 10 to 1,000.

```
n_estimators in [10, 100, 1000]
```

```
Out[54]: ▾                     RandomForestClassifier
         RandomForestClassifier(n_estimators=1000, random_state=42)
```



```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97      1708
           1       0.85      0.83      0.84       318

    accuracy                           0.95      2026
   macro avg       0.91      0.90      0.91      2026
weighted avg       0.95      0.95      0.95      2026
```



For Random Forest model we are getting accuracy of 95%

MODEL SUMMARY

After evaluating all the Models we can conclude that Random Forest is performing well in all metrics.

CONCLUSION

1. There are 16.07% of customers who have churned
2. Bank should focus on low income customers as from our analysis we concluded that they are more likely to attrited.
3. Blue card holder also have significantly higher attrition rate, Focusing on them will significantly improve attrition.
4. Churn prediction - Random Forest model created with around 95% accuracy to predict whether given customer will churn or not