# MACHINE LEARNING CASE STUDY

Telco Customer Churn Prediction

MARCH 2023

Ashley Bao
ashley_bao@outlook.com

# Introduction

Customer Churn means whether the customer has stopped using the product and switched to a competitor product. The Churn rate is one of the most important measures of customer satisfaction and operational performance for telecommunication companies. Research has shown that on average, it is 5 to 25 times more expensive to acquire a new customer than to retain an existing customer and a 5% increase in customer retention can lead to a revenue increase of 25% to 95%.

The objective is to accurately identify and predict customer churn and retention by applying the Machine Learning model with XGBoost, helping the company develop strategies to retain these clients.

The dataset was taken from a Kaggle challenge.

# Contents

## 1. Loading Libraries and Data

The first step is to load the necessary packages and import the dataset into pandas.

```
#import necessary packages

import numpy as np
import pandas as pd
pd.set_option('display.max_columns', 500)

import matplotlib.pyplot as plt
import seaborn as sns
import xgboost as xgb

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, roc_auc_score, make_scorer
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix

import warnings
warnings.filterwarnings('ignore')
```

```
#import dataset into pandas
df = pd.read_excel('Telco_customer_churn.xlsx')
```

## 2. Understanding the Data

The next step is to explore the dataset and grasp the idea of the data structure.

```
df.shape
```

```
]: (7043, 33)
```

```
df.columns.values
```

```
]: array(['CustomerID', 'Count', 'Country', 'State', 'City', 'Zip Code',
       'Lat Long', 'Latitude', 'Longitude', 'Gender', 'Senior Citizen',
       'Partner', 'Dependents', 'Tenure Months', 'Phone Service',
       'Multiple Lines', 'Internet Service', 'Online Security',
       'Online Backup', 'Device Protection', 'Tech Support',
       'Streaming TV', 'Streaming Movies', 'Contract',
       'Paperless Billing', 'Payment Method', 'Monthly Charges',
       'Total Charges', 'Churn Label', 'Churn Value', 'Churn Score',
       'CLTV', 'Churn Reason'], dtype=object)
```

```
df.head()
```

]:

| | CustomerID | Count | Country | State | City | Zip Code | Lat Long | Latitude | Longitude | Gender | Senior Citizen | Partner | Dependents | Tenure Months | Phone Service | Mu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 3668-QPYBK | 1 | United States | California | Los Angeles | 90003 | 33.964131, -118.272783 | 33.964131 | -118.272783 | Male | No | No | No | 2 | Yes | |
| 1 | 9237-HQITU | 1 | United States | California | Los Angeles | 90005 | 34.059281, -118.30742 | 34.059281 | -118.307420 | Female | No | No | Yes | 2 | Yes | |
| 2 | 9305-CDSKC | 1 | United States | California | Los Angeles | 90006 | 34.048013, -118.293953 | 34.048013 | -118.293953 | Female | No | No | Yes | 8 | Yes | |
| 3 | 7892-POOKP | 1 | United States | California | Los Angeles | 90010 | 34.062125, -118.315709 | 34.062125 | -118.315709 | Female | No | Yes | Yes | 28 | Yes | |
| 4 | 0280-XJGEX | 1 | United States | California | Los Angeles | 90015 | 34.039224, -118.266293 | 34.039224 | -118.266293 | Male | No | No | Yes | 49 | Yes | |

As shown above, the dataset contains 7043 rows and 33 columns, which includes information about:

1. Dependent Variable:

- Churn Label (1 = the customer left the company this quarter. 0 = the customer remained with the company. Directly related to Churn Value.)

*We will drop Churn Value(Yes or No), Churn Score, CLTV, CLTV, and Churn Reason during the modeling section.

2. Independent Variables:

- Demographic Information: Country, State, City, Zip Code, Lat Long, Latitude, Longitude, Gender, Senior Citizen, Partner, Dependents
- Customer Account Information: Tenure Months, Contract, Paperless Billing, Payment Method, Monthly Charges, Total Charges
- Services Information: Phone Service, Multiple Lines, Internet Service, Online Security, Online Backup, Device Protection, Tech Support, Streaming TV, Streaming Movies

By further exploring the dataset, we can understand the data type, unique values, and number of these unique values of each column, which are listed below.

```
def report(df):
    col = []
    d_type = []
    uniques = []
    n_uniques = []

    for i in df.columns:
        col.append(i)
        d_type.append(df[i].dtypes)
        uniques.append(df[i].unique()[:3])
        n_uniques.append(df[i].nunique())

    return pd.DataFrame({'Column': col, 'd_type': d_type, 'unique_sample': uniques, 'n_uniques': n_uniques})

pd.set_option('max.colwidth', 120)


report(df)
```

| | Column | d_type | unique_sample | n_uniques |
|---|---|---|---|---|
| **0** | CustomerID | object | [3668-QPYBK, 9237-HQITU, 9305-CDSKC] | 7043 |
| **1** | Count | int64 | [1] | 1 |
| **2** | Country | object | [United States] | 1 |
| **3** | State | object | [California] | 1 |
| **4** | City | object | [Los Angeles, Beverly Hills, Huntington Park] | 1129 |
| **5** | Zip Code | int64 | [90003, 90005, 90006] | 1652 |
| **6** | Lat Long | object | [33.964131, -118.272783, 34.059281, -118.30742, 34.048013, -118.293953] | 1652 |
| **7** | Latitude | float64 | [33.964131, 34.059281, 34.048013] | 1652 |
| **8** | Longitude | float64 | [-118.272783, -118.30742, -118.293953] | 1651 |
| **9** | Gender | object | [Male, Female] | 2 |
| **10** | Senior Citizen | object | [No, Yes] | 2 |
| **11** | Partner | object | [No, Yes] | 2 |
| **12** | Dependents | object | [No, Yes] | 2 |
| **13** | Tenure Months | int64 | [2, 8, 28] | 73 |
| **14** | Phone Service | object | [Yes, No] | 2 |
| **15** | Multiple Lines | object | [No, Yes, No phone service] | 3 |
| **16** | Internet Service | object | [DSL, Fiber optic, No] | 3 |
| **17** | Online Security | object | [Yes, No, No internet service] | 3 |
| **18** | Online Backup | object | [Yes, No, No internet service] | 3 |
| **19** | Device Protection | object | [No, Yes, No internet service] | 3 |
| **20** | Tech Support | object | [No, Yes, No internet service] | 3 |
| **21** | Streaming TV | object | [No, Yes, No internet service] | 3 |
| **22** | Streaming Movies | object | [No, Yes, No internet service] | 3 |
| **23** | Contract | object | [Month-to-month, Two year, One year] | 3 |
| **24** | Paperless Billing | object | [Yes, No] | 2 |
| **25** | Payment Method | object | [Mailed check, Electronic check, Bank transfer (automatic)] | 4 |
| **26** | Monthly Charges | float64 | [53.85, 70.7, 99.65] | 1585 |
| **27** | Total Charges | object | [108.15, 151.65, 820.5] | 6531 |
| **28** | Churn Label | object | [Yes, No] | 2 |
| **29** | Churn Value | int64 | [1, 0] | 2 |
| **30** | Churn Score | int64 | [86, 67, 84] | 85 |
| **31** | CLTV | int64 | [3239, 2701, 5372] | 3438 |
| **32** | Churn Reason | object | [Competitor made better offer, Moved, Competitor had better devices] | 20 |

## 3. Data Cleaning

After grasping the mail idea of the dataset, it is time to head to data pre-processing and data cleaning.

First, we will replace the space in between the column name with underscore. Second, from the data type report from the last step, we observe that the column “Total_Charges” was listed as an object. Since this column represents the total amount charged to the customer and it should be detected as numeric variable. Therefore, we need to transform this column into a numeric data type.

```
df.columns = df.columns.str.replace(' ','_', regex=True)
df.columns
```

```
Index(['CustomerID', 'Count', 'Country', 'State', 'City', 'Zip_Code',
       'Lat_Long', 'Latitude', 'Longitude', 'Gender', 'Senior_Citizen',
       'Partner', 'Dependents', 'Tenure_Months', 'Phone_Service',
       'Multiple_Lines', 'Internet_Service', 'Online_Security',
       'Online_Backup', 'Device_Protection', 'Tech_Support', 'Streaming_TV',
       'Streaming_Movies', 'Contract', 'Paperless_Billing', 'Payment_Method',
       'Monthly_Charges', 'Total_Charges', 'Churn_Label', 'Churn_Value',
       'Churn_Score', 'CLTV', 'Churn_Reason'],
      dtype='object')
```

```
# Converting Total Charges to a numerical data type
df['Total_Charges'] = pd.to_numeric(df['Total_Charges'], errors='coerce')
```

Next, by using the isnull() function, we are allowed to find out the missing value across the whole dataset. In the list below, there are 11 missing values in the column of “Total_Charges”.

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

```
CustomerID            0
Count                 0
Country               0
State                 0
City                  0
Zip_Code              0
Lat_Long              0
Latitude              0
Longitude             0
Gender                0
Senior_Citizen        0
Partner               0
Dependents            0
Tenure_Months         0
Phone_Service         0
Multiple_Lines        0
Internet_Service      0
Online_Security       0
Online_Backup         0
Device_Protection     0
Tech_Support          0
Streaming_TV          0
Streaming_Movies      0
Contract              0
Paperless_Billing     0
Payment_Method        0
Monthly_Charges       0
Total_Charges        11
Churn_Label           0
Churn_Value           0
Churn_Score           0
CLTV                  0
Churn_Reason       5174
dtype: int64
```

Since 11 is a relatively small number given the whole dataset contains 7043 observations. We decided to remove those observations from the dataset.

```
df.dropna(subset=['Total_Charges'],inplace=True)
```

In the meantime, we will drop other variables that do not help explain whether the customer will churn.

```
df.drop(columns=['CustomerID','Count','City','Country','Zip_Code','Lat_Long','State','Latitude','Longitude'], inplace=True)
```

And we will turn “Bank transfer (automatic)”, “Credit card (automatic)” to “Bank transfer” and “Credit card”, as these denominations are too long to be used as tick labels when we conduct the data visualization in the next section.

```
df['Payment_Method'] = df['Payment_Method'].str.replace(' (automatic)', '', regex=False)
```

```
df.Payment_Method.unique()
```

```
7]: array(['Mailed check', 'Electronic check', 'Bank transfer', 'Credit card'],
          dtype=object)
```

## 4. Data Visualization

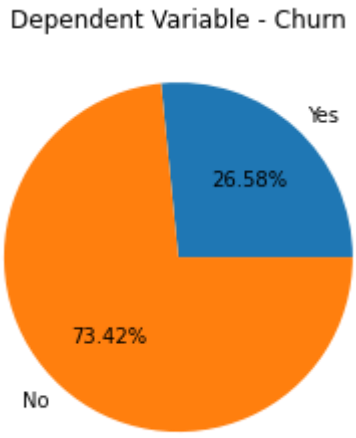
In this section, we will develop the exploratory data analysis by using visualization with matplotlib and seaborn.

a. Dependent Variables

We will start from the dependent variable which is “Churn_Label”.

As shown below, this is an imbalanced data set because both classes are not equally distributed among all observations. We will handle the imbalance data during the machine learning model section.

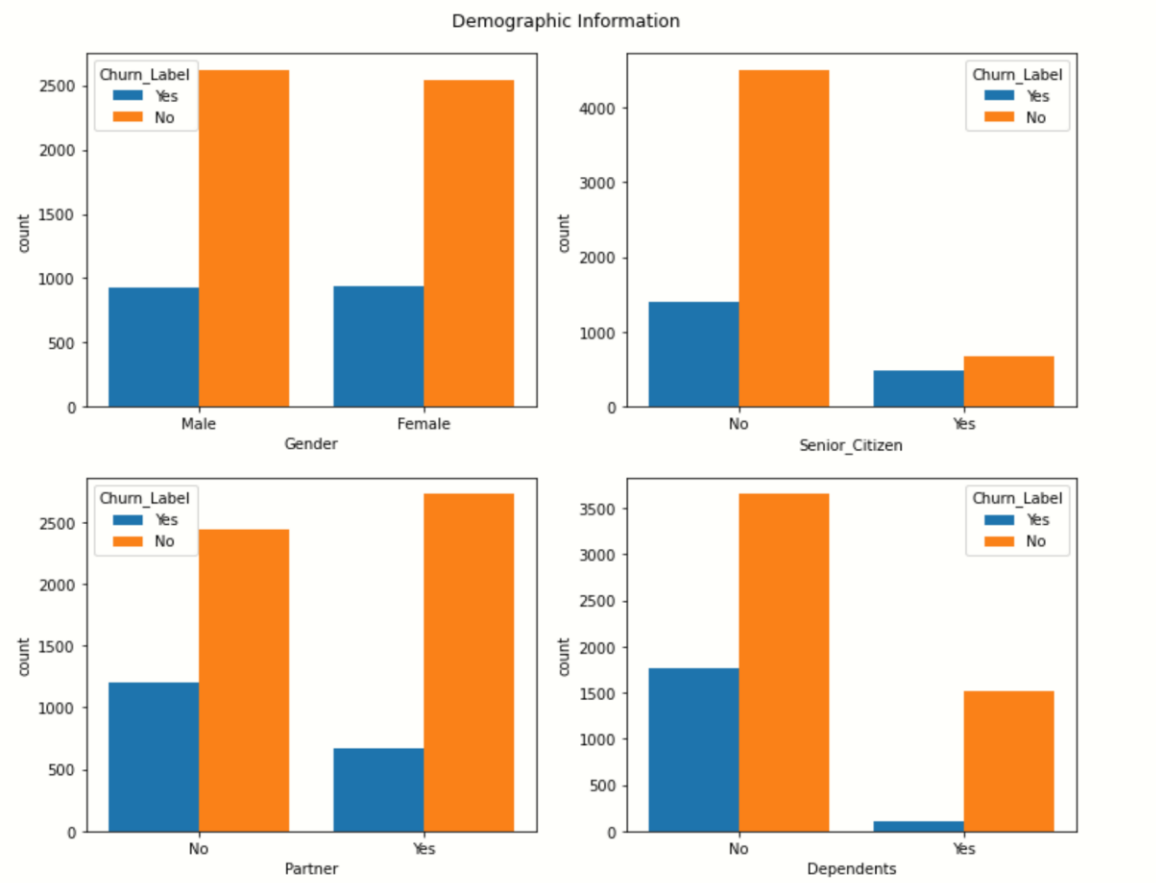
```
plt.pie(df['Churn_Label'].value_counts().sort_values(), labels=df['Churn_Label'].unique(), autopct='%.2f%%')
plt.title('Dependent Variable - Churn')
plt.show()
```



b. Demographic Information

We will start to explore the independent variables by plotting the demographic information.

```
fig, axes = plt.subplots(2, 2, figsize=(12,10))
fig.suptitle('Demographic Information', y=0.92)

#Gender
ax1 = sns.countplot(ax=axes[0,0], data=df, x='Gender', hue='Churn_Label')
#Senior Citizen
ax2 = sns.countplot(ax=axes[0,1], data=df, x='Senior_Citizen', hue='Churn_Label')
#Partner
ax3 = sns.countplot(ax=axes[1,0], data=df, x='Partner', hue='Churn_Label')
#Dependents
ax4 = sns.countplot(ax=axes[1,1], data=df, x='Dependents', hue='Churn_Label')

plt.show()
```

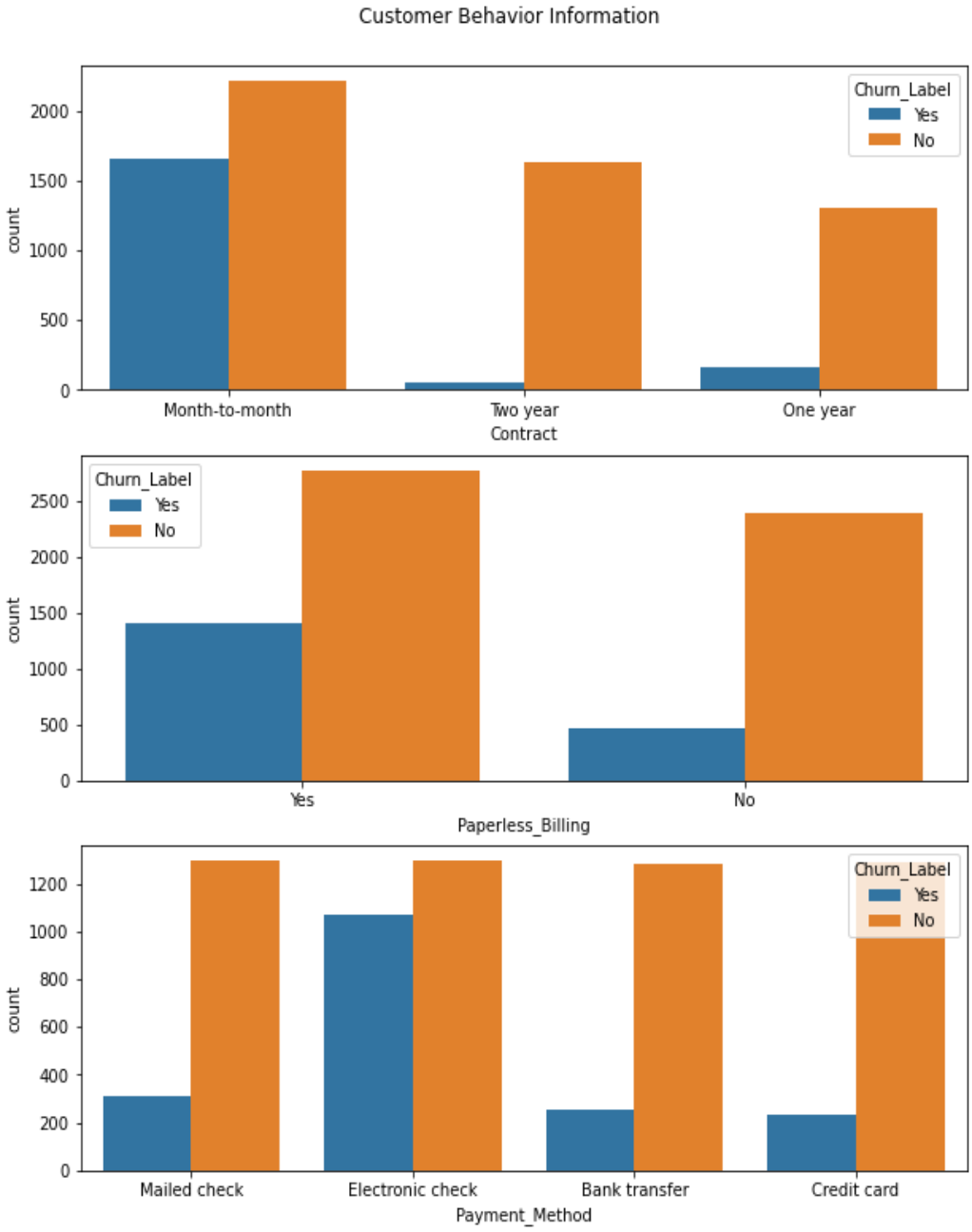



Conclusions:

- Gender does not have significant predictive power over the churn rate.
- The churn rate of senior citizens (65 or older) is higher than that of young citizens.
- Customers with a partner churn less than customers with no partner.
- Customers with a dependent churn less than customers with no dependent.

c. Customer Account Information - Categorical variables

```
fig, axes = plt.subplots(3, 1, figsize=(10,12))
fig.suptitle('Customer Behavior Information',y=0.92)

#Contract
sns.countplot(ax=axes[0], data=df, x='Contract', hue='Churn_Label')
#Paperless_Billing
sns.countplot(ax=axes[1], data=df, x='Paperless_Billing', hue='Churn_Label')
#Payment_Method
sns.countplot(ax=axes[2], data=df, x='Payment_Method', hue='Churn_Label')

plt.show()
```

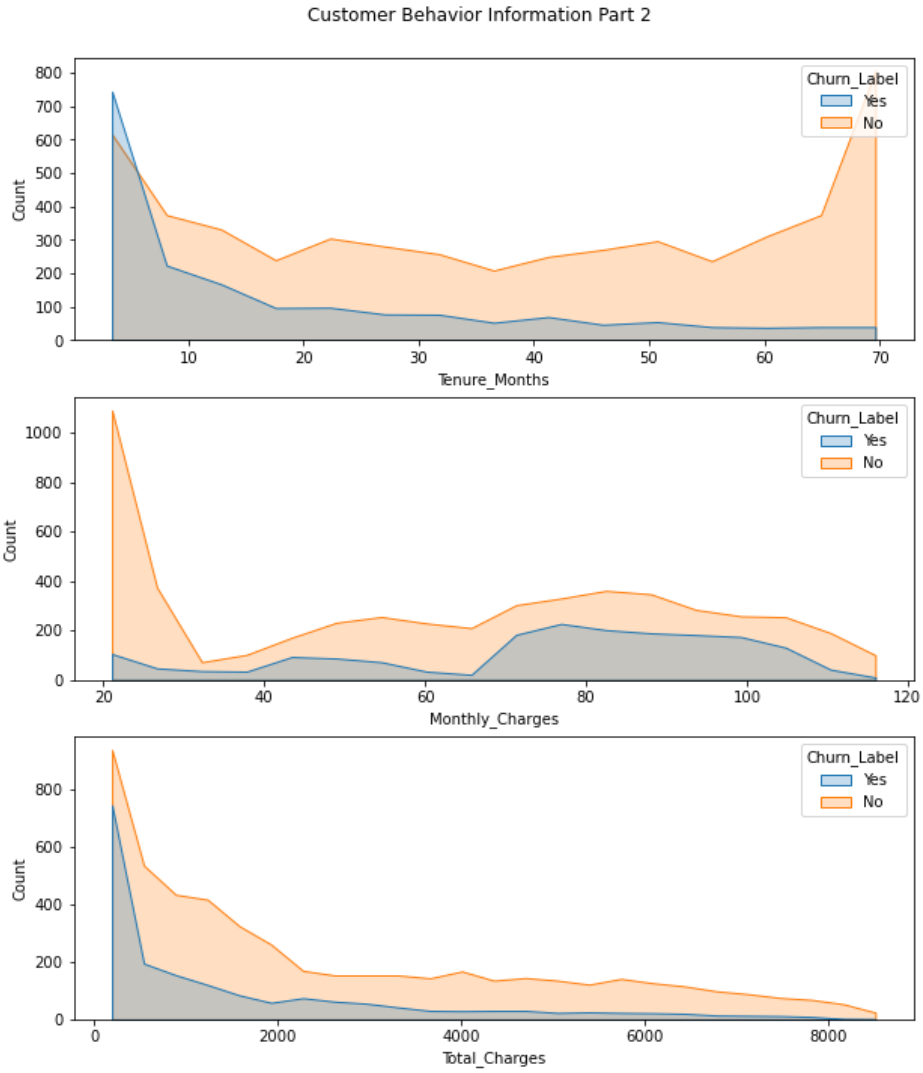


Conclusions:

- Customers with month-to-month contracts have higher churn rates compared to customers with yearly contracts.
- The churn rate of customers subscribed to paperless billing is almost double that of those who are not subscribed.
- Customers that chose an electronic check as the payment method are more likely to churn.

d. Customer Account Information - Numerical variables

```
fig, axes = plt.subplots(3, 1, figsize=(10,12))
fig.suptitle('Customer Behavior Information Part 2',y=0.92)

sns.histplot(ax=axes[0], data=df, x='Tenure_Months', hue='Churn_Label',element="poly")
sns.histplot(ax=axes[1], data=df, x='Monthly_Charges', hue='Churn_Label',element="poly")
sns.histplot(ax=axes[2], data=df, x='Total_Charges', hue='Churn_Label',element="poly")

plt.show()
```

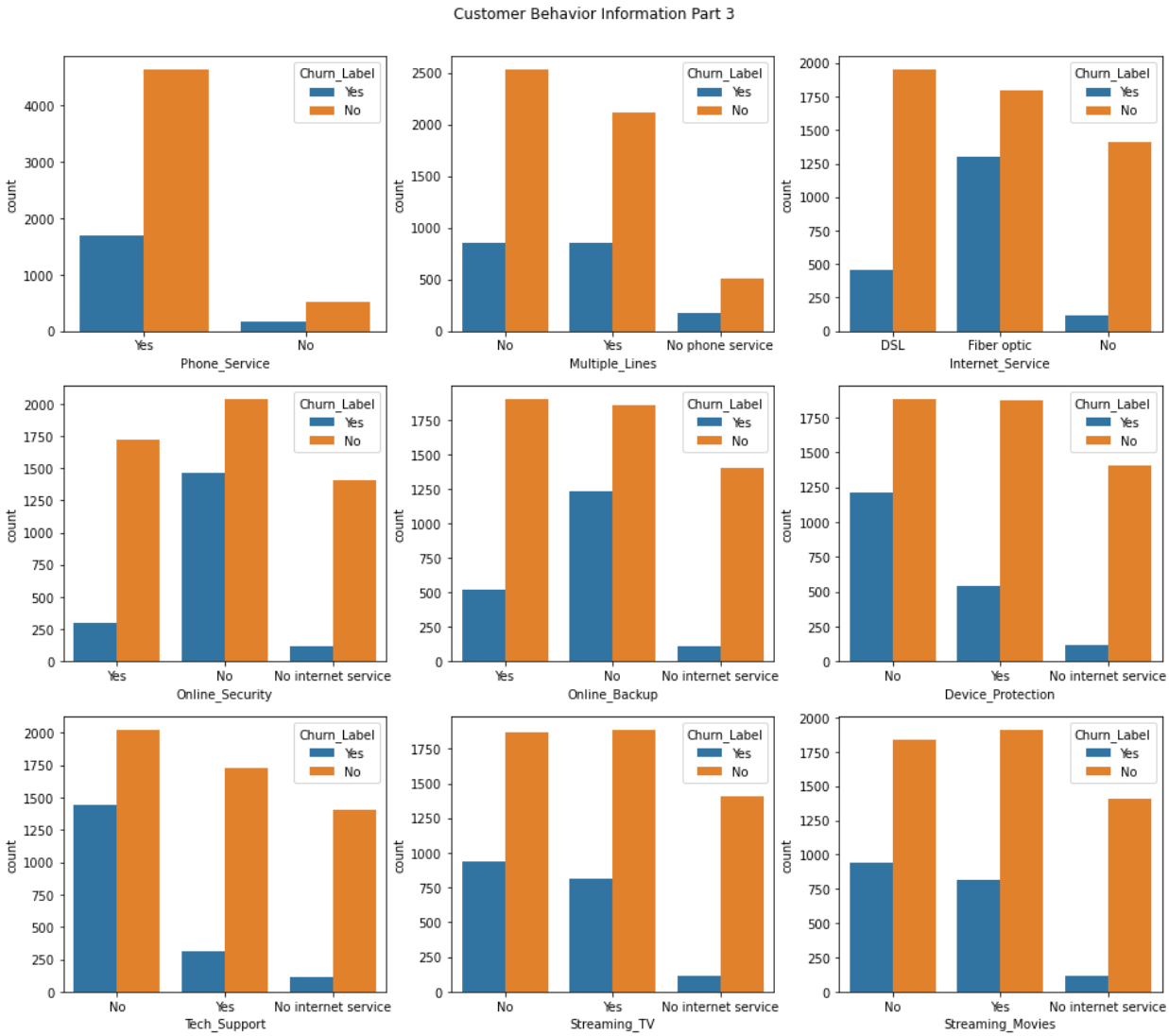


Conclusions:

- New customers (especially within 10 tenure months) are more likely to churn.
- The churn rate tends to be larger when monthly charges are high, especially when it is over $65 per month.
- The higher the total charges are, the fewer customers are likely to churn.

e. Services Information

```
services_columns = ['Phone_Service', 'Multiple_Lines', 'Internet_Service', 'Online_Security',
                    'Online_Backup', 'Device_Protection', 'Tech_Support', 'Streaming_TV', 'Streaming_Movies']

fig, axes = plt.subplots(3,3,figsize = (16,14))
fig.suptitle('Customer Behavior Information Part 3', y=0.92)

for i, item in enumerate(services_columns):
    if i < 3:
        sns.countplot(ax=axes[0,i], data=df, x=df[item], hue='Churn_Label')

    elif i >=3 and i < 6:
        sns.countplot(ax=axes[1,i-3], data=df, x=df[item], hue='Churn_Label')

    elif i < 9:
        sns.countplot(ax=axes[2,i-6], data=df, x=df[item], hue='Churn_Label')

plt.show()
```




Conclusions:

- Phone attributes such as phone service and multiple lines do not have significant predictive power.
- Customers who are subscribing to services like online security, online backup, tech support and device protection tend to churn less than those without these services.

## 5. Feature Engineering

One hot encoding is a process of converting categorical data variables so they can be provided to machine learning algorithms to improve predictions.

Before that, we need to drop the unrelative variables such as Churn Value, Churn Score, CLTV, CLTV, and Churn Reason as we have mentioned in the previous section.

```
df.drop(columns=['Churn_Label','Churn_Score','CLTV','Churn_Reason'], inplace=True)
df.head()
```

]:

| | Gender | Senior_Citizen | Partner | Dependents | Tenure_Months | Phone_Service | Multiple_Lines | Internet_Service | Online_Security | Online_Backup | Device_F |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Male | No | No | No | 2 | Yes | No | DSL | Yes | Yes | |
| 1 | Female | No | No | Yes | 2 | Yes | No | Fiber optic | No | No | |
| 2 | Female | No | No | Yes | 8 | Yes | Yes | Fiber optic | No | No | |
| 3 | Female | No | Yes | Yes | 28 | Yes | Yes | Fiber optic | No | No | |
| 4 | Male | No | No | Yes | 49 | Yes | Yes | Fiber optic | No | Yes | |

Then we will convert the following variables into numerical data from categorical data.

```
x = df.drop(['Churn_Value'], axis=1).copy()

x_encoded = pd.get_dummies(x,columns=['Gender',
                                      'Senior_Citizen',
                                      'Partner',
                                      'Dependents',
                                      'Phone_Service',
                                      'Multiple_Lines',
                                      'Internet_Service',
                                      'Online_Security',
                                      'Online_Backup',
                                      'Device_Protection',
                                      'Tech_Support',
                                      'Streaming_TV',
                                      'Streaming_Movies',
                                      'Contract',
                                      'Paperless_Billing',
                                      'Payment_Method'])
x_encoded.head()
```

]:

| | Tenure_Months | Monthly_Charges | Total_Charges | Gender_Female | Gender_Male | Senior_Citizen_No | Senior_Citizen_Yes | Partner_No | Partner_Yes | Dependc |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2 | 53.85 | 108.15 | 0 | 1 | 1 | 0 | 1 | 0 | |
| 1 | 2 | 70.70 | 151.65 | 1 | 0 | 1 | 0 | 1 | 0 | |
| 2 | 8 | 99.65 | 820.50 | 1 | 0 | 1 | 0 | 1 | 0 | |
| 3 | 28 | 104.80 | 3046.05 | 1 | 0 | 1 | 0 | 0 | 1 | |
| 4 | 49 | 103.70 | 5036.30 | 0 | 1 | 1 | 0 | 1 | 0 | |

## 6. Splitting the Data

We will split the data into training and testing sets. The stratify parameter will preserve the proportion of target as in original dataset, in the train and test datasets as well.

As shown below, the training set and testing set is split with the same ratio of the target variable, which is 0.266, by applying the stratify parameter.

```
x_train, x_test, y_train, y_test = train_test_split(x_encoded,y, random_state=42, stratify=y)
```

```
sum(y_train)/len(y_train)
```

```
55]: 0.2658323852863102
```

```
sum(y_test)/len(y_test)
```

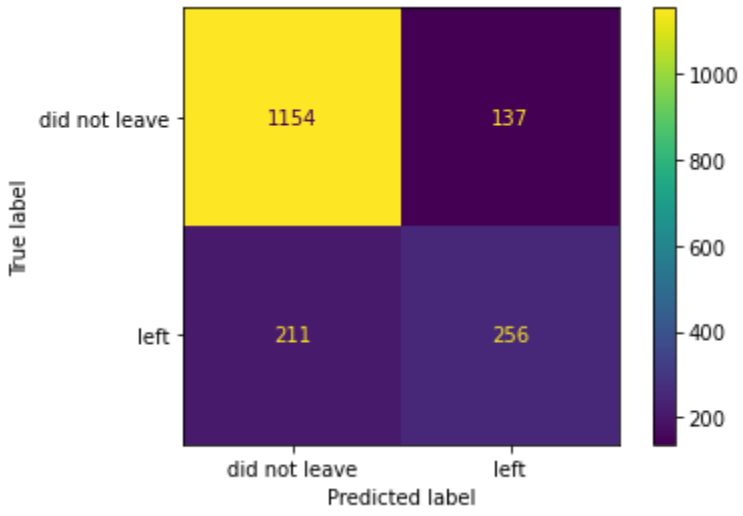
```
56]: 0.2656427758816837
```

## 7. Machine Learning Model Evaluations and Predictions

In this section, we will implement the XGBoost model. Since XGBoost is an ensemble algorithm comprised of decision trees, it does not require normalization for the inputs. Therefore, we will skip normalization procedure though which is a common practice in most of other machine learning practices.

```
clf_xgb= xgb.XGBClassifier(objective='binary:logistic',max_depth=5,learning_rate= 0.1,
                           colsample_bytree = 0.5,
                           subsample=0.90, reg_lambda= 10,use_label_encoder=False)

clf_xgb.fit(x_train,y_train,early_stopping_rounds=10,eval_metric='aucpr',eval_set=[(x_test,y_test)])
```

The Confusion Matrix is plotted below. The total accuracy is 72.1% with 89% recall (True Positive Rate) on people who did not leave and 55% recall rate on people who did leave.



```
print('The accuracy of the model is:',roc_auc_score(y_test,y_pred))
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
The accuracy of the model is: 0.7210302920731071
              precision    recall  f1-score   support

           0       0.85      0.89      0.87      1291
           1       0.65      0.55      0.60       467

    accuracy                           0.80      1758
   macro avg       0.75      0.72      0.73      1758
weighted avg       0.79      0.80      0.80      1758
```

Conclusion:

The model correctly classified 89% of people who did not leave the company. However, it does not perform well with people who had left the company with only 55% correctly classified, which means there were 467 people churned but only 256 of them were identified.

For telecommunication companies, the churn rate is very important, especially recognizing customers who churn is more significant than identifying those who do not.

Therefore, in order to improve accuracy on recognizing customers who churn, we will find out the best performance parameter by using Cross-Validation with Grid Search technique. In addition, XGBoost has a parameter “scale_pos_weight” that helps with imbalanced data.

## 8. Hyperparameter Tuning with GridSearchCV

As shown below, the best parameters are gamma=1.0, learning_rate=0.1, max_depth=3, reg_lambda=10, scale_pos_weight=3.

```
param_grid={'max_depth':[3,4,5],
            'gamma':[0,0.25,1.0],
            'learning_rate':[0.1,0.5,1.0],
            'reg_lambda':[1,10,20],
            'scale_pos_weight':[1,3,5]}
```

```
optimal_param= GridSearchCV(estimator=xgb.XGBClassifier(objective='binary:logistic',seed=42,
                                                        subsample=0.9,colsample_bytree=0.5,use_label_encoder=False),
                            param_grid=param_grid,
                            scoring='roc_auc',
                            verbose=2,
                            cv=5)
```

```
optimal_param.fit(x_train,y_train,early_stopping_rounds=10,eval_metric='auc',eval_set=[(x_test,y_test)],verbose=False)
```

```
Fitting 5 folds for each of 243 candidates, totalling 1215 fits
```

```
print('The optimal parameters are:', optimal_param.best_params_)
```

```
The optimal parameters are: {'gamma': 1.0, 'learning_rate': 0.1, 'max_depth': 3, 'reg_lambda': 10, 'scale_pos_weight': 3}
```

## 9. Evaluating the Final Model

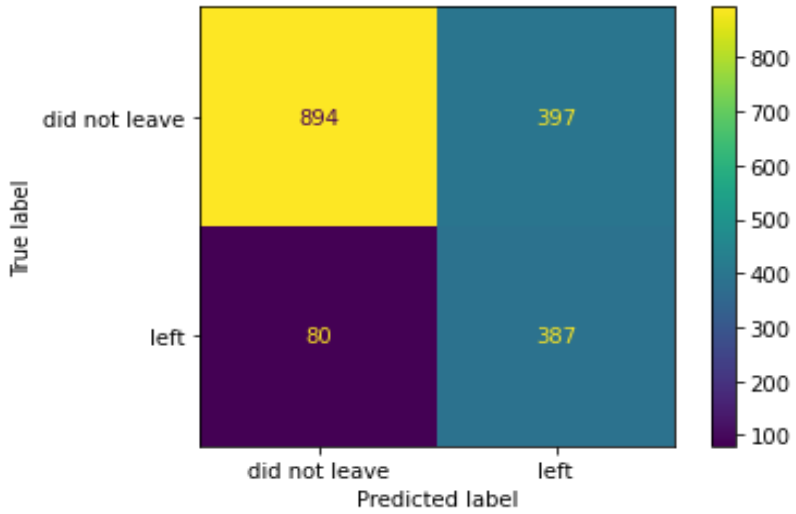
Once the best performance parameters are finally generated, we can apply them to the model and reevaluate the performance.

```
#rebuild the model with the best parameters

clf_classifier= xgb.XGBClassifier(objective='binary:logistic',
                                  gamma=1.0,
                                  learning_rate=0.1,
                                  max_depth=3,
                                  reg_lambda=10,
                                  scale_pos_weight=3,
                                  subsample=0.9,
                                  colsample_bytree=0.50,
                                  use_label_encoder=False)
```

```
clf_classifier.fit(x_train,y_train,early_stopping_rounds=10,eval_metric='aucpr',eval_set=[(x_test,y_test)])
```

```
y_pred= clf_classifier.predict(x_test)
plot_confusion_matrix(clf_classifier,x_test,y_test,values_format='d',display_labels=['did not leave','left'])
```

The Confusion Matrix shows there is a significant improvement in the prediction accuracy on the people who did churn by increasing the recall rate from 55% to 83%.




```
print('The accuracy of the model:',roc_auc_score(y_test,y_pred))
```

```
The accuracy of the model: 0.7605901173832347
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.92      0.69      0.79      1291
           1       0.49      0.83      0.62       467

    accuracy                           0.73      1758
   macro avg       0.71      0.76      0.70      1758
weighted avg       0.81      0.73      0.74      1758
```

Conclusion:

After applying the best parameters, the model performed better with people who had left the company by increasing the recall rate from 55% to 83% even though the recall rate for people who did not churn dropped to 69%.

Since identifying people who churn is much important than who does not churn in this project, the final model after the Hyperparameter Tuning provide a better result which can help the company to spot and develop strategies accordingly to retain these clients.