Author: Dimgba Martha Otisi

@martha_samuel_

This code uses Logistic Regression model to predict Churn of a Company.

In [1]:
```
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
import pickle
```

In [2]:
```
df= pd.read_csv('Train.csv')
print(df.head(15))
```

```
                                     user_id       REGION          TEN
URE  \
0   dcf68cc2fb515ccad7d8b9b3bd80ee2a4b270063  SAINT-LOUIS   K > 24 mo
nth
1   71c44b5ba328db5c4192a80f7cf8f244d9350ed0          NaN   K > 24 mo
nth
2   ce46411b1526c94f20a383b8cb188f8d27f82a0a  TAMBACOUNDA   K > 24 mo
nth
3   f467cdb6669818373c26c2bad44e01ba66f97d21       FATICK   K > 24 mo
nth
4   ec45e1a1888a32b5dcce0954cfec20c6e037db31       FATICK   K > 24 mo
nth
5   2bd9ab2983615149380a63f44a66780f4fa19a4a       THIES   K > 24 mo
nth
6   b2d9c4bdceaafe305e8424c97f64e4ba880d0a97          NaN   K > 24 mo
nth
7   8ebce4e82fa049f96ff1aa460217171af4e4ede1  SAINT-LOUIS  H 15-18 mo
nth
8   ebfbd28870a7663d49ec79799f9fd59e8c5655ed  TAMBACOUNDA   K > 24 mo
```

In [3]:
```
df_Test = pd.read_csv('Test.csv')
print(df_Test.head())
```

```
                                    user_id REGION        TENURE  MONT
ANT  \
0  af900d87e73b7ff6509d2203df4704a98aa5f2a6    NaN  K > 24 month      
NaN
1  5335efd940280b82143272275637d1e65d37eadb    NaN  K > 24 month      
NaN
2  a581f4fa08677c26f83f643248c667e241043086    NaN  K > 24 month   190
0.0
3  64f67177d0775262b8087a9e2e3b8061b6324ae6  DAKAR  K > 24 month   300
0.0
4  0d6009a4594c4be22449b8d9cc01a0bcea98faea  DAKAR  K > 24 month  3200
0.0

   FREQUENCE_RECH  REVENUE  ARPU_SEGMENT  FREQUENCE  DATA_VOLUME  ON_N
ET  \
0             NaN      NaN           NaN        NaN          NaN     N
aN
1             NaN     10.0           3.0        1.0          NaN     N
aN
2            15.0   2299.0         766.0       21.0        414.0     N
aN
3             9.0   2603.0         868.0       14.0        332.0      
0.0
4            47.0  33000.0       11000.0       47.0          NaN   12
8.0

   ORANGE   TIGO  ZONE1  ZONE2 MRG  REGULARITY               TOP_PACK 
\
0     NaN    NaN    NaN    NaN  NO           1                    NaN 
1     NaN    NaN    NaN    NaN  NO           2                    NaN 
2     7.0    2.0    NaN    NaN  NO          27   Data: 100 F=40MB,24H 
3    23.0    4.0    NaN    NaN  NO          46    IVR Echat_Daily_50F 
4   555.0  280.0    NaN    NaN  NO          61  All-net 500F=2000F;5d 

   FREQ_TOP_PACK
0            NaN
1            NaN
2           17.0
3            3.0
4           65.0
```

In [4]:
```
print(df.shape)
print(df_Test.shape)
```

```
(400000, 19)
(100000, 18)
```

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400000 entries, 0 to 399999
Data columns (total 19 columns):
 #   Column          Non-Null Count   Dtype  
---  ------          --------------   -----  
 0   user_id         400000 non-null  object 
 1   REGION          242480 non-null  object 
 2   TENURE          400000 non-null  object 
 3   MONTANT         259723 non-null  float64
 4   FREQUENCE_RECH  259723 non-null  float64
 5   REVENUE         265337 non-null  float64
 6   ARPU_SEGMENT    265337 non-null  float64
 7   FREQUENCE       265337 non-null  float64
 8   DATA_VOLUME     203146 non-null  float64
 9   ON_NET          254181 non-null  float64
 10  ORANGE          233683 non-null  float64
 11  TIGO            160614 non-null  float64
 12  ZONE1           31690 non-null   float64
 13  ZONE2           25513 non-null   float64
 14  MRG             400000 non-null  object 
 15  REGULARITY      400000 non-null  int64  
 16  TOP_PACK        232671 non-null  object 
 17  FREQ_TOP_PACK   232671 non-null  float64
 18  CHURN           400000 non-null  int64  
dtypes: float64(12), int64(2), object(5)
memory usage: 58.0+ MB
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['user_id', 'REGION', 'TENURE', 'MONTANT', 'FREQUENCE_RECH', 'RE
VENUE',
       'ARPU_SEGMENT', 'FREQUENCE', 'DATA_VOLUME', 'ON_NET', 'ORANGE',
'TIGO',
       'ZONE1', 'ZONE2', 'MRG', 'REGULARITY', 'TOP_PACK', 'FREQ_TOP_PA
CK',
       'CHURN'],
      dtype='object')
```

In [7]:
```
df_Test.describe()
```

Out[7]:

	MONTANT	FREQUENCE_RECH	REVENUE	ARPU_SEGMENT	FREQUENCE	DATA
count	65049.000000	65049.000000	66510.000000	66510.000000	66510.000000	50
mean	5545.613630	11.545051	5518.341663	1839.453676	13.979973	3
std	7123.955226	13.271270	7177.840304	2392.609422	14.655983	10
min	25.000000	1.000000	1.000000	0.000000	1.000000	
25%	1000.000000	2.000000	1000.000000	333.000000	3.000000	
50%	3000.000000	7.000000	3000.000000	1000.000000	9.000000	
75%	7400.000000	16.000000	7399.000000	2466.000000	20.000000	2
max	201500.000000	120.000000	181135.000000	60378.000000	91.000000	474

In [8]:
```
df.describe()
```

Out[8]:

	MONTANT	FREQUENCE_RECH	REVENUE	ARPU_SEGMENT	FREQUENCE	DAT
count	259723.000000	259723.000000	265337.000000	265337.000000	265337.000000	20
mean	5522.971346	11.503733	5505.487757	1835.167658	13.951835	
std	7099.640630	13.275514	7175.802367	2391.929290	14.679943	1
min	20.000000	1.000000	1.000000	0.000000	1.000000	
25%	1000.000000	2.000000	1000.000000	333.000000	3.000000	
50%	3000.000000	6.000000	3000.000000	1000.000000	9.000000	
75%	7300.000000	15.000000	7340.000000	2447.000000	19.000000	
max	226550.000000	133.000000	233413.000000	77804.000000	91.000000	93

In [9]:
```
df.isna().sum()#checks for number of missing data
```

Out[9]:
```
user_id                 0
REGION             157520
TENURE                  0
MONTANT            140277
FREQUENCE_RECH     140277
REVENUE            134663
ARPU_SEGMENT       134663
FREQUENCE          134663
DATA_VOLUME        196854
ON_NET             145819
ORANGE             166317
TIGO               239386
ZONE1              368310
ZONE2              374487
MRG                     0
REGULARITY              0
TOP_PACK           167329
FREQ_TOP_PACK      167329
CHURN                   0
dtype: int64
```

In [10]:
```
df_Test.isna().sum()
```

Out[10]:
```
user_id                0
REGION             39293
TENURE                 0
MONTANT            34951
FREQUENCE_RECH     34951
REVENUE            33490
ARPU_SEGMENT       33490
FREQUENCE          33490
DATA_VOLUME        49338
ON_NET             36383
ORANGE             41200
TIGO               59788
ZONE1              92320
ZONE2              93578
MRG                    0
REGULARITY             0
TOP_PACK           41703
FREQ_TOP_PACK      41703
dtype: int64
```

In [11]:
```
df_user = df.pop('user_id')
#df.drop(['user_id'], 1, inplace = True)
```

In [12]:
```
df_user_Test = df_Test.pop('user_id')
```

In [13]:
```
def handle_non_numerical_data(df):
    columns = df.columns.values

    for column in columns:
        text_digit_vals = {}#Eg here we could have {'NO' : 0, etc}, it c
        def convert_to_int(val):
            return text_digit_vals[val]

        if df[column].dtype != np.int64 and df[column].dtype != np.float
            column_contents = df[column].values.tolist()#we convert to l
            unique_elements = set(column_contents)#then take the unique 
            x=0
            for unique in unique_elements:
                if unique not in text_digit_vals:
                    text_digit_vals[unique] = x
                    x+=1

            df[column] = list(map(convert_to_int, df[column]))#we reset 
            #values in first parameter to the 2nd parameter
    return df
df = handle_non_numerical_data(df)
print(df.head(15))
df_Test = handle_non_numerical_data(df_Test)
print(df_Test.head())
```

```
    REGION  TENURE  MONTANT  FREQUENCE_RECH  REVENUE  ARPU_SEGMENT  FR
EQUENCE  \
0       14       5  17000.0            32.0  18000.0        6000.0
34.0
1        0       5   4300.0            29.0   4427.0        1476.0
37.0
2        6       5   1500.0             3.0   1500.0         500.0
3.0
3        8       5   1500.0             3.0   2497.0         832.0
4.0
4        8       5      NaN             NaN    498.0         166.0
3.0
5        3       5   5500.0             9.0   5359.0        1786.0
19.0
6        0       5      NaN             NaN      NaN           NaN
NaN
7       14       7      NaN             NaN      NaN           NaN
NaN
8        6       5  22500.0             8.0  22230.0        7410.0
16.0
9        0       6      NaN             NaN      NaN           NaN
NaN
10       3       5      NaN             NaN      NaN           NaN
NaN
11       3       5    300.0             3.0    300.0         100.0
3.0
12      13       5   3300.0             9.0   2996.0         999.0
10.0
13       9       5   4550.0            18.0   4450.0        1483.0
23.0
14       9       5   3800.0             7.0   3500.0        1167.0
```

```
7.0

    DATA_VOLUME  ON_NET  ORANGE   TIGO  ZONE1  ZONE2  MRG  REGULARITY  \
0           NaN    97.0   355.0    6.0    NaN    NaN    0          62
1        1764.0     8.0     3.0    0.0    NaN    2.0    0          40
2           NaN    30.0    30.0    NaN    NaN    NaN    0          32
3           0.0   159.0    45.0   19.0    NaN    NaN    0          18
4           1.0     1.0     3.0    NaN    NaN    NaN    0          50
5        6084.0     7.0    12.0    5.0    NaN    NaN    0          30
6           NaN     NaN     NaN    NaN    NaN    NaN    0           7
7           0.0     NaN     NaN    NaN    NaN    NaN    0           5
8       14956.0  6336.0  1017.0  185.0    NaN    NaN    0          62
9           NaN     NaN     NaN    NaN    NaN    NaN    0           1
10          NaN     NaN     NaN    NaN    NaN    NaN    0           3
11        169.0     8.0     1.0    0.0    NaN    NaN    0          47
12          0.0   547.0     9.0    0.0    NaN    NaN    0          54
13       1062.0    34.0    20.0    4.0    NaN    NaN    0          58
14          NaN    15.0    71.0    5.0    NaN    NaN    0          24

    TOP_PACK  FREQ_TOP_PACK  CHURN
0          1           35.0      0
1         61           22.0      0
2          1            3.0      0
3         79            3.0      0
4          0            NaN      0
5         22            7.0      0
6          0            NaN      0
7          0            NaN      1
8         49            3.0      0
9          0            NaN      0
10         0            NaN      0
11       102            2.0      0
12        79            4.0      0
13        16           11.0      0
14        29            1.0      0
   REGION  TENURE  MONTANT  FREQUENCE_RECH  REVENUE  ARPU_SEGMENT  FRE
QUENCE  \
0       0       5      NaN             NaN      NaN           NaN
NaN
1       0       5      NaN             NaN     10.0           3.0
1.0
2       0       5   1900.0            15.0   2299.0         766.0
21.0
3       9       5   3000.0             9.0   2603.0         868.0
14.0
4       9       5  32000.0            47.0  33000.0       11000.0
47.0

   DATA_VOLUME  ON_NET  ORANGE   TIGO  ZONE1  ZONE2  MRG  REGULARITY
\
0          NaN     NaN     NaN    NaN    NaN    NaN    0           1
1          NaN     NaN     NaN    NaN    NaN    NaN    0           2
2        414.0     NaN     7.0    2.0    NaN    NaN    0          27
3        332.0     0.0    23.0    4.0    NaN    NaN    0          46
4          NaN   128.0   555.0  280.0    NaN    NaN    0          61
```

```
   TOP_PACK  FREQ_TOP_PACK
0         0            NaN
1         0            NaN
2        50           17.0
3        27            3.0
4         1           65.0
```

In [14]:
```
df["CHURN"].value_counts().values
```

Out[14]:
```
array([325156,  74844])
```

In [15]:
```
df.shape
```

Out[15]:
```
(400000, 18)
```

In [16]:
```
df.describe()
```

Out[16]:

	REGION	TENURE	MONTANT	FREQUENCE_RECH	REVENUE	ARPU_
count	400000.000000	400000.000000	259723.000000	259723.000000	265337.000000	265
mean	4.983425	4.937597	5522.971346	11.503733	5505.487757	1
std	4.884023	0.590715	7099.640630	13.275514	7175.802367	2
min	0.000000	0.000000	20.000000	1.000000	1.000000	
25%	0.000000	5.000000	1000.000000	2.000000	1000.000000	
50%	3.000000	5.000000	3000.000000	6.000000	3000.000000	1
75%	9.000000	5.000000	7300.000000	15.000000	7340.000000	2
max	14.000000	7.000000	226550.000000	133.000000	233413.000000	77

In [17]:
```
df_Test.describe()
```

Out[17]:

	REGION	TENURE	MONTANT	FREQUENCE_RECH	REVENUE	ARPU_
count	100000.000000	100000.000000	65049.000000	65049.000000	66510.000000	66
mean	4.986220	4.936360	5545.613630	11.545051	5518.341663	1
std	4.885387	0.594116	7123.955226	13.271270	7177.840304	2
min	0.000000	0.000000	25.000000	1.000000	1.000000	
25%	0.000000	5.000000	1000.000000	2.000000	1000.000000	
50%	3.000000	5.000000	3000.000000	7.000000	3000.000000	1
75%	9.000000	5.000000	7400.000000	16.000000	7399.000000	2
max	14.000000	7.000000	201500.000000	120.000000	181135.000000	60

In [18]:
```
df.isnull()
```

Out[18]:

	REGION	TENURE	MONTANT	FREQUENCE_RECH	REVENUE	ARPU_SEGMENT	FRE
0	False	False	False	False	False	False	
1	False	False	False	False	False	False	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	True	True	False	False	
...	...	...	...	...	...	...	
399995	False	False	False	False	False	False	
399996	False	False	True	True	True	True	
399997	False	False	False	False	False	False	
399998	False	False	True	True	False	False	
399999	False	False	True	True	True	True	

400000 rows × 18 columns

In [19]:
```
#this replaces NaN with the mean of its column
column_means = df.mean()
df=df.fillna(column_means)
print(df.head(15))
```

```
    REGION  TENURE       MONTANT  FREQUENCE_RECH       REVENUE  ARPU_S
EGMENT  \
0       14       5  17000.000000       32.000000  18000.000000   6000.
000000
1        0       5   4300.000000       29.000000   4427.000000   1476.
000000
2        6       5   1500.000000        3.000000   1500.000000    500.
000000
3        8       5   1500.000000        3.000000   2497.000000    832.
000000
4        8       5   5522.971346       11.503733    498.000000    166.
000000
5        3       5   5500.000000        9.000000   5359.000000   1786.
000000
6        0       5   5522.971346       11.503733   5505.487757   1835.
167658
7       14       7   5522.971346       11.503733   5505.487757   1835.
167658
8        6       5  22500.000000        8.000000  22230.000000   7410.
000000
9        0       6   5522.971346       11.503733   5505.487757   1835.
167658
10       3       5   5522.971346       11.503733   5505.487757   1835.
167658
11       3       5    300.000000        3.000000    300.000000    100.
000000
12      13       5   3300.000000        9.000000   2996.000000    999.
000000
13       9       5   4550.000000       18.000000   4450.000000   1483.
000000
14       9       5   3800.000000        7.000000   3500.000000   1167.
000000

    FREQUENCE   DATA_VOLUME       ON_NET       ORANGE        TIGO
ZONE1  \
0   34.000000   3369.763441    97.000000   355.000000    6.000000   7.8
74282
1   37.000000   1764.000000     8.000000     3.000000    0.000000   7.8
74282
2    3.000000   3369.763441    30.000000    30.000000   23.134608   7.8
74282
3    4.000000      0.000000   159.000000    45.000000   19.000000   7.8
74282
4    3.000000      1.000000     1.000000     3.000000   23.134608   7.8
74282
5   19.000000   6084.000000     7.000000    12.000000    5.000000   7.8
74282
6   13.951835   3369.763441   275.917586    95.532927   23.134608   7.8
74282
7   13.951835      0.000000   275.917586    95.532927   23.134608   7.8
74282
8   16.000000  14956.000000  6336.000000  1017.000000  185.000000   7.8
```

```
74282
9   13.951835   3369.763441   275.917586    95.532927   23.134608  7.8
74282
10  13.951835   3369.763441   275.917586    95.532927   23.134608  7.8
74282
11   3.000000    169.000000     8.000000     1.000000    0.000000  7.8
74282
12  10.000000      0.000000   547.000000     9.000000    0.000000  7.8
74282
13  23.000000   1062.000000    34.000000    20.000000    4.000000  7.8
74282
14   7.000000   3369.763441    15.000000    71.000000    5.000000  7.8
74282

       ZONE2  MRG  REGULARITY  TOP_PACK  FREQ_TOP_PACK  CHURN
0   7.187003    0          62         1      35.000000      0
1   2.000000    0          40        61      22.000000      0
2   7.187003    0          32         1       3.000000      0
3   7.187003    0          18        79       3.000000      0
4   7.187003    0          50         0       9.254209      0
5   7.187003    0          30        22       7.000000      0
6   7.187003    0           7         0       9.254209      0
7   7.187003    0           5         0       9.254209      1
8   7.187003    0          62        49       3.000000      0
9   7.187003    0           1         0       9.254209      0
10  7.187003    0           3         0       9.254209      0
11  7.187003    0          47       102       2.000000      0
12  7.187003    0          54        79       4.000000      0
13  7.187003    0          58        16      11.000000      0
14  7.187003    0          24        29       1.000000      0
```

In [20]:
```
column_means = df_Test.mean()
df_Test=df_Test.fillna(column_means)
print(df_Test.head(5))
```

```
   REGION  TENURE      MONTANT  FREQUENCE_RECH       REVENUE  ARPU_SEG
MENT  \
0       0       5   5545.61363       11.545051   5518.341663   1839.45
3676
1       0       5   5545.61363       11.545051     10.000000      3.00
0000
2       0       5   1900.00000       15.000000   2299.000000    766.00
0000
3       9       5   3000.00000        9.000000   2603.000000    868.00
0000
4       9       5  32000.00000       47.000000  33000.000000  11000.00
0000

   FREQUENCE  DATA_VOLUME      ON_NET      ORANGE        TIGO      ZONE
1  \
0  13.979973  3357.428033  279.370703   94.900799   23.459291  8.37330
7
1   1.000000  3357.428033  279.370703   94.900799   23.459291  8.37330
7
2  21.000000   414.000000  279.370703    7.000000    2.000000  8.37330
7
3  14.000000   332.000000    0.000000   23.000000    4.000000  8.37330
7
4  47.000000  3357.428033  128.000000  555.000000  280.000000  8.37330
7

      ZONE2  MRG  REGULARITY  TOP_PACK  FREQ_TOP_PACK
0  7.678138    0           1         0       9.276035
1  7.678138    0           2         0       9.276035
2  7.678138    0          27        50      17.000000
3  7.678138    0          46        27       3.000000
4  7.678138    0          61         1      65.000000
```

In [21]:
```
#Create Feature variable X and Target variable y

y = np.array(df['CHURN'])
```

In [22]:
```
X = np.array(df.drop(['CHURN'], 1).astype(float))
#Standardizing/scaling the features
X = StandardScaler().fit_transform(X)
print(len(X),len(y))
```

```
400000 400000
```

In [23]:
```
x = np.array((df_Test).astype(float))
x = StandardScaler().fit_transform(x)
print(len(x))
```

```
100000
```

In [24]:
```
#Split the data into 80% training and 20% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =0.2
# using logistic regression model
clf = LogisticRegression()
clf.fit(X_train, y_train)
```

Out[24]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept
=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verb
ose=0,
                   warm_start=False)
```

In [25]:
```
# evaluating the model
prediction = clf.predict(X_test)
#printing the predictions
print(prediction)
print(len(prediction))
```

```
[0 0 1 ... 0 1 0]
80000
```

In [26]:
```
#Generate confusion matrix for logistics regression model as it has maxi

conf_mat_clf = confusion_matrix(y_test,prediction)
conf_mat_clf
```

Out[26]:
```
array([[59012,  5855],
       [ 4217, 10916]])
```

In [27]:
```
# Predict the probability of Churn of each customer
predictions = clf.predict(X)
print(predictions)
print(len(predictions))
```

```
[0 0 0 ... 0 0 1]
400000
```

In [28]:
```
import sklearn
sklearn.metrics.accuracy_score(prediction,y_test)
```

Out[28]:
```
0.8741
```

In [29]:
```
# Finding precision and recall
from sklearn.metrics import precision_score, recall_score
precision_score(prediction, y_test)
recall_score(prediction, y_test)
```

Out[29]:
```
0.6508854570389363
```

In [30]:
```
from sklearn.metrics import log_loss
log_loss(prediction, y_test) # the smaller the logloss(uncertainty), the
```

Out[30]: 4.348474096898794

In [31]:
```
'''for pickle'''
with open ('logisticregression.pickle','wb') as f:
    pickle.dump(clf, f)
```

In [32]:
```
'''we use this cell while testing on new data after we have written pick
'''to read the pickle'''
pickle_in = open('logisticregression.pickle','rb')
'''we renamed classifier here'''
clf = pickle.load(pickle_in)
```

In [33]:
```
solution = clf.predict(x)
print(solution)
print(len(solution))
```

```
[1 1 0 ... 0 0 0]
100000
```

In [34]:
```
df4 = pd.DataFrame(solution) #converting prediction to a dataframe
df4.rename(columns={0: "CHURN"})  #rename the column from '0' to 'CHURN'
```

Out[34]:

	CHURN
0	1
1	1
2	0
3	0
4	0
...	...
99995	0
99996	0
99997	0
99998	0
99999	0

100000 rows × 1 columns

In [35]:
```
# Predict the probability of Churn of each customer
df['Churn'] = predictions
#print(df['Churn'])
print(len(df['Churn']))
#print(df['CHURN'])
df2 = pd.DataFrame(df['Churn'], columns = ['Churn'])
print(df2.tail(20))
```

```
400000
        Churn
399980      0
399981      0
399982      0
399983      0
399984      0
399985      0
399986      0
399987      0
399988      0
399989      0
399990      1
399991      1
399992      0
399993      0
399994      0
399995      0
399996      0
399997      0
399998      0
399999      1
```

In [36]:
```
# Predict the probability of Churn of each customer
'''df['CHURN'] = pd.Series(solution) #fixed valueerror

print(len(df['CHURN']))
print(df['CHURN'])
df_Test = pd.DataFrame(df['CHURN'], columns = ['CHURN'])
df4 = df_Test.iloc[0:100000]
print(df4.tail(20))
print(len(df4))'''
```

Out[36]:
```
"df['CHURN'] = pd.Series(solution) #fixed valueerror\n\nprint(len(df
['CHURN']))\nprint(df['CHURN'])\ndf_Test = pd.DataFrame(df['CHURN'], c
olumns = ['CHURN'])\ndf4 = df_Test.iloc[0:100000]\nprint(df4.tail(20))
\nprint(len(df4))"
```

In [37]:
```
print(df_user)
df1 = pd.DataFrame(df_user, columns = ['user_id'])
print(df1.tail(20))
```

```
0         dcf68cc2fb515ccad7d8b9b3bd80ee2a4b270063
1         71c44b5ba328db5c4192a80f7cf8f244d9350ed0
2         ce46411b1526c94f20a383b8cb188f8d27f82a0a
3         f467cdb6669818373c26c2bad44e01ba66f97d21
4         ec45e1a1888a32b5dcce0954cfec20c6e037db31
                            ...
399995    a892ad4ed0eda8dc721733200c47147763b183ec
399996    13daa3a651bf0192a413b339c4766aeafc6d1636
399997    767f596aee426962f7d92f4de8d7b232cdc17568
399998    b831e4d3b59a1e294e9e0a2aab391bc12d50845c
399999    a80e3a164986e489102cfb538fa62e16ecc28adf
Name: user_id, Length: 400000, dtype: object
                                         user_id
399980  7b902e808901aceb41731abfb82ac7ec856ca452
399981  186963eda12e9f661af8ce952b36929554de69e7
399982  2f9211523724ec0a6f9da554f1364dc07d49b89e
399983  e6464f573cdcfc8d92e3d0c3f96a0925515d9fe8
399984  f64d8938c3d1ce0401b4d7af4c94a6f8e1a8c85d
399985  a63a501d82db5227001c57fd67fcffb08bcc0fc0
399986  200e23c4639f968fec5b21dadc31cbae61499c1e
399987  e94e33eccf6397d2d111b5eda98c3f00d775957c
399988  0d5ee6f1c0499866f8742351691a29984f79a953
399989  e52db82cb925cdcc6f7cd4bae73b8aa2fb48c383
399990  991f37c6c994daab8ab8c07a2453a43ebb060f53
399991  2a92663555f9c237a0e69462625e8916b8182d5e
399992  6efd8294140f2e410b8cbbbafa77c2b096ca35a5
399993  a708001683c19272a0f7648a61f832a99ae8a3d5
399994  6979ea052669404eae2f58a90df8fd482640915d
399995  a892ad4ed0eda8dc721733200c47147763b183ec
399996  13daa3a651bf0192a413b339c4766aeafc6d1636
399997  767f596aee426962f7d92f4de8d7b232cdc17568
399998  b831e4d3b59a1e294e9e0a2aab391bc12d50845c
399999  a80e3a164986e489102cfb538fa62e16ecc28adf
```

In [38]:
```
print(df_user_Test)
df3 = pd.DataFrame(df_user_Test, columns = ['user_id'])
print(df3.tail(20))
```

```
0        af900d87e73b7ff6509d2203df4704a98aa5f2a6
1        5335efd940280b82143272275637d1e65d37eadb
2        a581f4fa08677c26f83f643248c667e241043086
3        64f67177d0775262b8087a9e2e3b8061b6324ae6
4        0d6009a4594c4be22449b8d9cc01a0bcea98faea
                           ...
99995    c6bcb3336795a18eb6c0bc7e19078a0704ef4d7e
99996    a44b4e44dc70115ed5bf971ebb4193dd536e87f0
99997    a2f84faffbc995bd0e2d726fa4ffdb93f11646ed
99998    afa76e894df4201fc77eb714de7d1f262299611a
99999    c08a2d84b87c1f5d4bb318114f508b77aa8e2663
Name: user_id, Length: 100000, dtype: object
                                        user_id
99980  501dbe56ea737f87a8c21d711e954bb4f36f2cc3
99981  24838af1f3fbcb95b19ba14e0e977a037cd91bf0
99982  552a68f1fb242bf1c645495fbf7528399050ba0b
99983  7cc53deb9749ff830caa9a5ea1cdb1b21c634540
99984  7bda478c0f7229565bd4d937fcfd53ce37dd71df
99985  64808d2cf540ace5dc242632c75329ad92c558e4
```

In [46]:
```

print(len(X),len(y))
result = pd.concat([df1, df2], axis=1)

print(result)

```

```
400000 400000
                                         user_id  Churn
0       dcf68cc2fb515ccad7d8b9b3bd80ee2a4b270063      0
1       71c44b5ba328db5c4192a80f7cf8f244d9350ed0      0
2       ce46411b1526c94f20a383b8cb188f8d27f82a0a      0
3       f467cdb6669818373c26c2bad44e01ba66f97d21      0
4       ec45e1a1888a32b5dcce0954cfec20c6e037db31      0
...                                          ...    ...
399995  a892ad4ed0eda8dc721733200c47147763b183ec      0
399996  13daa3a651bf0192a413b339c4766aeafc6d1636      0
399997  767f596aee426962f7d92f4de8d7b232cdc17568      0
399998  b831e4d3b59a1e294e9e0a2aab391bc12d50845c      0
399999  a80e3a164986e489102cfb538fa62e16ecc28adf      1

[400000 rows x 2 columns]
```

In [165]:
```
result.to_csv('Churn_predict_file')#to text file
```

In [65]:
```
df5= pd.read_csv('sample_submission.csv')
df5.drop(['CHURN'], axis=1, inplace=True)
Test_result=pd.concat([df5,df4], axis=1)
print(Test_result)
print(len(df5),len(df4))
```

```
                                        user_id  0
0      af900d87e73b7ff6509d2203df4704a98aa5f2a6  1
1      5335efd940280b82143272275637d1e65d37eadb  1
2      a581f4fa08677c26f83f643248c667e241043086  0
3      64f67177d0775262b8087a9e2e3b8061b6324ae6  0
4      0d6009a4594c4be22449b8d9cc01a0bcea98faea  0
...                                         ... ..
99995  c6bcb3336795a18eb6c0bc7e19078a0704ef4d7e  0
99996  a44b4e44dc70115ed5bf971ebb4193dd536e87f0  0
99997  a2f84faffbc995bd0e2d726fa4ffdb93f11646ed  0
99998  afa76e894df4201fc77eb714de7d1f262299611a  0
99999  c08a2d84b87c1f5d4bb318114f508b77aa8e2663  0

[100000 rows x 2 columns]
100000 100000
```

In [167]:
```
Test_result.to_csv('Eprediction.csv', index=False)
```

In [168]:
```
Test_result
```

Out[168]:

	user_id	0
0	af900d87e73b7ff6509d2203df4704a98aa5f2a6	1
1	5335efd940280b82143272275637d1e65d37eadb	1
2	a581f4fa08677c26f83f643248c667e241043086	0
3	64f67177d0775262b8087a9e2e3b8061b6324ae6	0
4	0d6009a4594c4be22449b8d9cc01a0bcea98faea	0
...	...	...
99995	c6bcb3336795a18eb6c0bc7e19078a0704ef4d7e	0
99996	a44b4e44dc70115ed5bf971ebb4193dd536e87f0	0
99997	a2f84faffbc995bd0e2d726fa4ffdb93f11646ed	0
99998	afa76e894df4201fc77eb714de7d1f262299611a	0
99999	c08a2d84b87c1f5d4bb318114f508b77aa8e2663	0

100000 rows × 2 columns

In [ ]: