In [1]:
```
import pandas as pd
import numpy as np
import pickle
import warnings
warnings.simplefilter(action="ignore",category=FutureWarning)
warnings.simplefilter(action="ignore",category=UserWarning)
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import KFold,StratifiedKFold,train_test_split
from sklearn.metrics import roc_auc_score,accuracy_score,confusion_matrix,roc_curve,
```

In [2]:
```
a=pd.read_csv("C:\\Users\\reshma_koduri\\OneDrive\\Documents\\churn_prediction.csv")
a
```

Out[2]:

	customer_id	vintage	age	gender	dependents	occupation	city	customer_nw_category
0	1	3135	66	Male	0.0	self_employed	187.0	2
1	2	310	35	Male	0.0	self_employed	NaN	2
2	4	2356	31	Male	0.0	salaried	146.0	2
3	5	478	90	NaN	NaN	self_employed	1020.0	2
4	6	2531	42	Male	2.0	self_employed	1494.0	3
...	...	...	...	...	...	...	...	...
28377	30297	1845	10	Female	0.0	student	1020.0	2
28378	30298	4919	34	Female	0.0	self_employed	1046.0	2
28379	30299	297	47	Male	0.0	salaried	1096.0	2
28380	30300	2585	50	Male	3.0	self_employed	1219.0	3
28381	30301	2349	18	Male	0.0	student	1232.0	2

28382 rows × 21 columns

In [3]:
```
a.head(5)
```

Out[3]:

	customer_id	vintage	age	gender	dependents	occupation	city	customer_nw_category	bra
0	1	3135	66	Male	0.0	self_employed	187.0	2	
1	2	310	35	Male	0.0	self_employed	NaN	2	
2	4	2356	31	Male	0.0	salaried	146.0	2	
3	5	478	90	NaN	NaN	self_employed	1020.0	2	
4	6	2531	42	Male	2.0	self_employed	1494.0	3	

5 rows × 21 columns

In [4]:
```
a.tail(5)
```

Out[4]:

	customer_id	vintage	age	gender	dependents	occupation	city	customer_nw_category
28377	30297	1845	10	Female	0.0	student	1020.0	2
28378	30298	4919	34	Female	0.0	self_employed	1046.0	2
28379	30299	297	47	Male	0.0	salaried	1096.0	2
28380	30300	2585	50	Male	3.0	self_employed	1219.0	3
28381	30301	2349	18	Male	0.0	student	1232.0	2

5 rows × 21 columns

In [5]:
```
a.describe()
```

Out[5]:

	customer_id	vintage	age	dependents	city	customer_nw_category
count	28382.000000	28382.000000	28382.000000	25919.000000	27579.000000	28382.000000
mean	15143.508667	2364.336446	48.208336	0.347236	796.109576	2.225530
std	8746.454456	1610.124506	17.807163	0.997661	432.872102	0.660443
min	1.000000	180.000000	1.000000	0.000000	0.000000	1.000000
25%	7557.250000	1121.000000	36.000000	0.000000	409.000000	2.000000
50%	15150.500000	2018.000000	46.000000	0.000000	834.000000	2.000000
75%	22706.750000	3176.000000	60.000000	0.000000	1096.000000	3.000000
max	30301.000000	12899.000000	90.000000	52.000000	1649.000000	3.000000

In [6]:
```
a.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28382 entries, 0 to 28381
Data columns (total 21 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   customer_id                     28382 non-null  int64
 1   vintage                         28382 non-null  int64
 2   age                             28382 non-null  int64
 3   gender                          27857 non-null  object
 4   dependents                      25919 non-null  float64
 5   occupation                      28302 non-null  object
 6   city                            27579 non-null  float64
 7   customer_nw_category            28382 non-null  int64
 8   branch_code                     28382 non-null  int64
 9   days_since_last_transaction     25159 non-null  float64
 10  current_balance                 28382 non-null  float64
 11  previous_month_end_balance      28382 non-null  float64
 12  average_monthly_balance_prevQ   28382 non-null  float64
 13  average_monthly_balance_prevQ2  28382 non-null  float64
 14  current_month_credit            28382 non-null  float64
 15  previous_month_credit           28382 non-null  float64
```

```
 16  current_month_debit             28382 non-null  float64
 17  previous_month_debit            28382 non-null  float64
 18  current_month_balance           28382 non-null  float64
 19  previous_month_balance          28382 non-null  float64
 20  churn                           28382 non-null  int64
dtypes: float64(13), int64(6), object(2)
memory usage: 4.5+ MB
```

In [7]:
```
a['gender'].unique()
```

Out[7]:
```
array(['Male', nan, 'Female'], dtype=object)
```

In [8]:
```
a['gender'].value_counts()
```

Out[8]:
```
Male      16548
Female    11309
Name: gender, dtype: int64
```

In [9]:
```
a['dependents'].value_counts()
```

Out[9]:
```
0.0     21435
2.0      2150
1.0      1395
3.0       701
4.0       179
5.0        41
6.0         8
7.0         3
9.0         1
52.0        1
36.0        1
50.0        1
8.0         1
25.0        1
32.0        1
Name: dependents, dtype: int64
```

In [10]:
```
a['gender']=a['gender'].map({'Male':1,'Female':0})
a['gender']
```

Out[10]:
```
0        1.0
1        1.0
2        1.0
3        NaN
4        1.0
        ...
28377    0.0
28378    0.0
28379    1.0
28380    1.0
28381    1.0
Name: gender, Length: 28382, dtype: float64
```

In [11]:
```
a.isna().sum()
```

Out[11]:
```
customer_id                  0
vintage                      0
age                          0
gender                     525
dependents                2463
```

```
occupation                        80
city                             803
customer_nw_category               0
branch_code                        0
days_since_last_transaction     3223
current_balance                    0
previous_month_end_balance         0
average_monthly_balance_prevQ      0
average_monthly_balance_prevQ2     0
current_month_credit               0
previous_month_credit              0
current_month_debit                0
previous_month_debit               0
current_month_balance              0
previous_month_balance             0
churn                              0
dtype: int64
```

In [22]:
```
a.fillna(35,inplace=True)
```

In [23]:
```
#a['city']=a['city'].fillna(46)
```

In [24]:
```
a.head(20)
```

Out[24]:

	customer_id	vintage	age	gender	dependents	occupation	city	customer_nw_category	b
0	1	3135	66	1.0	0.0	self_employed	187.0	2	
1	2	310	35	1.0	0.0	self_employed	46.0	2	
2	4	2356	31	1.0	0.0	salaried	146.0	2	
3	5	478	90	35.0	35.0	self_employed	1020.0	2	
4	6	2531	42	1.0	2.0	self_employed	1494.0	3	
5	7	263	42	0.0	0.0	self_employed	1096.0	2	
6	8	5922	72	1.0	0.0	retired	1020.0	1	
7	9	1145	46	1.0	0.0	self_employed	623.0	2	
8	10	2132	31	1.0	0.0	salaried	1096.0	2	
9	11	3379	40	1.0	3.0	self_employed	1020.0	2	
10	12	661	68	1.0	0.0	retired	409.0	3	
11	13	7108	32	1.0	0.0	salaried	1096.0	1	
12	14	2438	73	1.0	0.0	retired	44.0	3	
13	15	5703	50	1.0	0.0	salaried	409.0	1	
14	16	2314	48	0.0	0.0	self_employed	665.0	2	
15	17	1934	51	0.0	0.0	self_employed	1232.0	3	
16	19	2723	49	1.0	0.0	self_employed	1125.0	3	
17	20	6111	52	0.0	0.0	self_employed	1096.0	2	
18	21	5821	47	0.0	1.0	self_employed	146.0	1	
19	22	3500	41	0.0	0.0	self_employed	1020.0	2	

20 rows × 21 columns

In [25]:
```
a.isna().sum()
```

Out[25]:
```
customer_id                     0
vintage                         0
age                             0
gender                          0
dependents                      0
occupation                      0
city                            0
customer_nw_category            0
branch_code                     0
days_since_last_transaction     0
current_balance                 0
previous_month_end_balance      0
average_monthly_balance_prevQ   0
average_monthly_balance_prevQ2  0
current_month_credit            0
previous_month_credit           0
current_month_debit             0
previous_month_debit            0
current_month_balance           0
previous_month_balance          0
churn                           0
dtype: int64
```

In [26]:
```
b=a.drop(['occupation','dependents','city','customer_id','branch_code','customer_nw_
b
```

Out[26]:

	vintage	age	days_since_last_transaction	current_balance	previous_month_end_balance	avera
0	3135	66	224.0	1458.71	1458.71	
1	310	35	60.0	5390.37	8704.66	
2	2356	31	35.0	3913.16	5815.29	
3	478	90	147.0	2291.91	2291.91	
4	2531	42	58.0	927.72	1401.72	
...	...	...	...	...	...	
28377	1845	10	70.0	1076.43	1076.43	
28378	4919	34	14.0	3844.10	4069.21	
28379	297	47	0.0	65511.97	61017.55	
28380	2585	50	35.0	1625.55	1625.55	
28381	2349	18	59.0	2107.05	2821.34	

28382 rows × 14 columns

In [27]:
```
c=pd.get_dummies(b,dtype=int)
c
```

Out[27]:

	vintage	age	days_since_last_transaction	current_balance	previous_month_end_balance	avera
0	3135	66	224.0	1458.71	1458.71	
1	310	35	60.0	5390.37	8704.66	
2	2356	31	35.0	3913.16	5815.29	
3	478	90	147.0	2291.91	2291.91	
4	2531	42	58.0	927.72	1401.72	
...	...	...	...	...	...	
28377	1845	10	70.0	1076.43	1076.43	
28378	4919	34	14.0	3844.10	4069.21	
28379	297	47	0.0	65511.97	61017.55	
28380	2585	50	35.0	1625.55	1625.55	
28381	2349	18	59.0	2107.05	2821.34	

28382 rows × 14 columns

In [28]:

```
y=c['churn']
y
```

Out[28]:

```
0        0
1        0
2        0
3        1
4        1
        ..
28377    0
28378    0
28379    1
28380    0
28381    1
Name: churn, Length: 28382, dtype: int64
```

In [29]:

```
x=c.drop(['churn'],axis=1)
x
```

Out[29]:

	vintage	age	days_since_last_transaction	current_balance	previous_month_end_balance	avera
0	3135	66	224.0	1458.71	1458.71	
1	310	35	60.0	5390.37	8704.66	
2	2356	31	35.0	3913.16	5815.29	
3	478	90	147.0	2291.91	2291.91	
4	2531	42	58.0	927.72	1401.72	
...	...	...	...	...	...	
28377	1845	10	70.0	1076.43	1076.43	
28378	4919	34	14.0	3844.10	4069.21	
28379	297	47	0.0	65511.97	61017.55	

	vintage	age	days_since_last_transaction	current_balance	previous_month_end_balance	avera
28380	2585	50	35.0	1625.55	1625.55	
28381	2349	18	59.0	2107.05	2821.34	

28382 rows × 13 columns

In [30]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
```

In [31]:
```
from sklearn.linear_model import LogisticRegression
reg=LogisticRegression()
reg.fit(x_train,y_train)
```

Out[31]:
```
LogisticRegression()
```

In [32]:
```
ypred=reg.predict(x_test)
ypred
```

Out[32]:
```
array([0, 0, 0, ..., 0, 1, 0], dtype=int64)
```

In [33]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,ypred)
```

Out[33]:
```
array([[7499,  146],
       [1332,  390]], dtype=int64)
```

In [34]:
```
from sklearn.metrics import accuracy_score
accuracy_score(ypred,y_test)
```

Out[34]:
```
0.8422120209245223
```

In [35]:
```
results=pd.DataFrame(columns=['actual','predicted'])
results['actual']=y_test
results['predicted']=ypred
results=results.reset_index()
results['Id']=results.index
results.head(10)
```

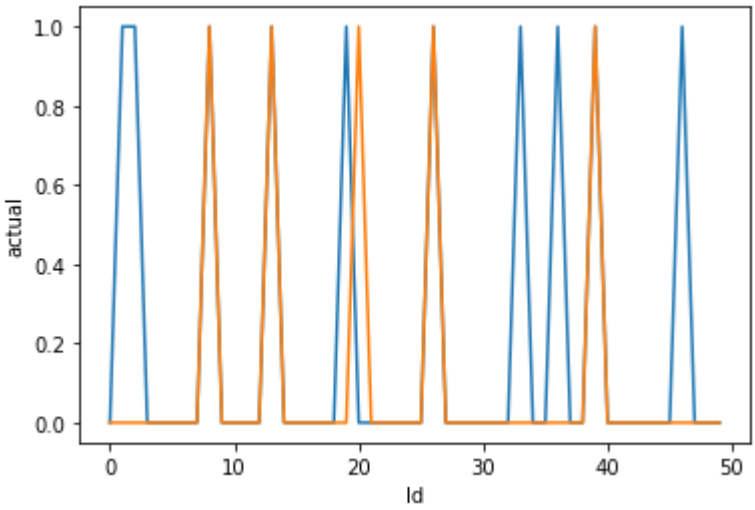
Out[35]:

	index	actual	predicted	Id
0	27546	0	0	0
1	16516	1	0	1
2	11680	1	0	2
3	20270	0	0	3
4	9185	0	0	4
5	19437	0	0	5
6	15344	0	0	6

	index	actual	predicted	Id
7	8665	0	0	7
8	2201	1	1	8
9	25571	0	0	9

In [36]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='Id',y='actual',data=results.head(50))
sns.lineplot(x='Id',y='predicted',data=results.head(50))
plt.plot()
```

Out[36]:
```
[]
```



In [43]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import KFold, cross_val_score
k_folds = KFold(n_splits = 4)
scores = cross_val_score(reg, x, y, cv = k_folds)
```

In [44]:
```
scores
```

Out[44]:
```
array([0.83850056, 0.82243517, 0.83326286, 0.83650458])
```

In [ ]:

In [ ]:

In [ ]: