In [1]:
```
import numpy as np
import pandas as pd
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
from sklearn.metrics import roc_auc_score, accuracy_score, confusion_matri
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.simplefilter(action='ignore', category=UserWarning)
```

In [2]:
```
df=pd.read_csv("churn_prediction.csv")
```

In [3]:
```
df
```

Out[3]:

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

28382 rows × 21 columns

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
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 [5]:
```
df['gender'].value_counts()
```

Out[5]:
```
gender
Male      16548
Female    11309
Name: count, dtype: int64
```

In [6]:
```
#here converting male to 1 and female as 0 and filling nul values as -1

dict_gender={'Male':1,'Female':0}
df.replace({'gender':dict_gender}, inplace=True)
df['gender']= df['gender'].fillna(-1)
```

In [7]:

```
df
```

Out[7]:

	customer_id	vintage	age	gender	dependents	occupation	city	customer_nw
0	1	3135	66	1.0	0.0	self_employed	187.0	
1	2	310	35	1.0	0.0	self_employed	NaN	
2	4	2356	31	1.0	0.0	salaried	146.0	
3	5	478	90	-1.0	NaN	self_employed	1020.0	
4	6	2531	42	1.0	2.0	self_employed	1494.0	
...	...	...	...	...	...	...	...	
28377	30297	1845	10	0.0	0.0	student	1020.0	
28378	30298	4919	34	0.0	0.0	self_employed	1046.0	
28379	30299	297	47	1.0	0.0	salaried	1096.0	
28380	30300	2585	50	1.0	3.0	self_employed	1219.0	
28381	30301	2349	18	1.0	0.0	student	1232.0	

28382 rows × 21 columns

In [8]:

```
df['dependents'].value_counts()
```

Out[8]:

```
dependents
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: count, dtype: int64
```

In [9]:

```
df['dependents']=df['dependents'].fillna(0)
df['occupation']=df['occupation'].fillna('self_employed')
df['city']=df['city'].fillna(1080)
df['days_since_last_transaction']=df['days_since_last_transaction'].fillna
```

In [10]:

```
df
```

Out[10]:

	customer_id	vintage	age	gender	dependents	occupation	city	customer_nw
0	1	3135	66	1.0	0.0	self_employed	187.0	
1	2	310	35	1.0	0.0	self_employed	1080.0	
2	4	2356	31	1.0	0.0	salaried	146.0	
3	5	478	90	-1.0	0.0	self_employed	1020.0	
4	6	2531	42	1.0	2.0	self_employed	1494.0	
...	...	...	...	...	...	...	...	
28377	30297	1845	10	0.0	0.0	student	1020.0	
28378	30298	4919	34	0.0	0.0	self_employed	1046.0	
28379	30299	297	47	1.0	0.0	salaried	1096.0	
28380	30300	2585	50	1.0	3.0	self_employed	1219.0	
28381	30301	2349	18	1.0	0.0	student	1232.0	

28382 rows × 21 columns

In [11]:

```
df = pd.concat([df,pd.get_dummies (df['occupation'], prefix = str('occupat
df
```

Out[11]:

	customer_id	vintage	age	gender	dependents	occupation	city	customer_nw
0	1	3135	66	1.0	0.0	self_employed	187.0	
1	2	310	35	1.0	0.0	self_employed	1080.0	
2	4	2356	31	1.0	0.0	salaried	146.0	
3	5	478	90	-1.0	0.0	self_employed	1020.0	
4	6	2531	42	1.0	2.0	self_employed	1494.0	
...	...	...	...	...	...	...	...	
28377	30297	1845	10	0.0	0.0	student	1020.0	
28378	30298	4919	34	0.0	0.0	self_employed	1046.0	
28379	30299	297	47	1.0	0.0	salaried	1096.0	
28380	30300	2585	50	1.0	3.0	self_employed	1219.0	
28381	30301	2349	18	1.0	0.0	student	1232.0	

28382 rows × 26 columns

Scaling Numerica Features for logistic Regression

since there a lot of outliers in dataset(prev,current bal).distrubutions are skewed so to deal it (Log transformation,Standard Scaler)

In [13]:
```
num_cols = ['customer_nw_category', 'current_balance',
            'previous_month_end_balance', 'average_monthly_balance_prevQ2'
            'current_month_credit','previous_month_credit', 'current_month
            'previous_month_debit','current_month_balance', 'previous_mont
for i in num_cols:
    df[i] = np.log(df[i] + 17000)

std = StandardScaler()
scaled = std.fit_transform(df[num_cols])
scaled = pd.DataFrame(scaled,columns=num_cols)
```

Model Evaluation'

Recall(to know wrongly marked)

AUC(area under curve),(X axis, ROC (reciever operating curve)

In [14]:
```
df_df_og = df.copy()
df = df.drop(columns = num_cols,axis = 1)
df = df.merge(scaled,left_index=True,right_index=True,how = "left")
```

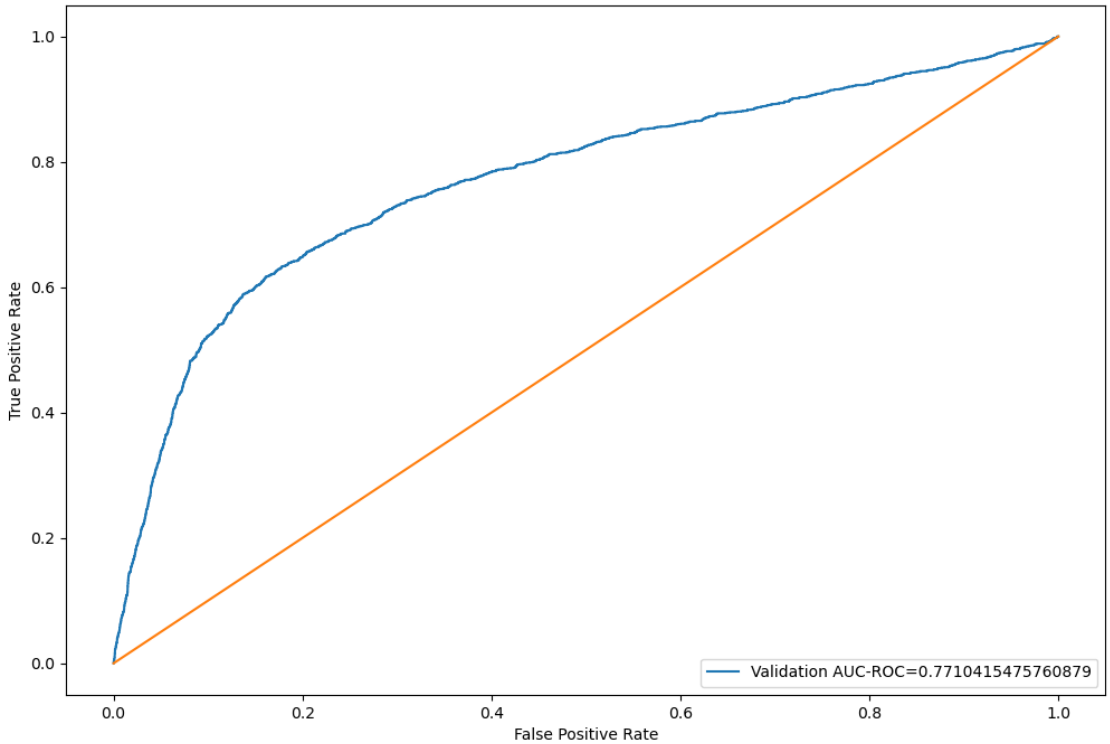
In [15]:
```
y_all = df.churn
df = df.drop(['churn','customer_id','occupation'],axis = 1)
```

In [16]:
```
baseline_cols = ['current_month_debit', 'previous_month_debit','current_ba
                 ,'occupation_retired', 'occupation_salaried','occupation_
df_baseline = df[baseline_cols]
```

In [19]:
```
from sklearn.model_selection import train_test_split
xtrain, xtest, ytrain, ytest = train_test_split(df_baseline,y_all,test_si
```

In [20]:
```
model = LogisticRegression()
model.fit(xtrain,ytrain)
pred = model.predict_proba(xtest)[:,1]
```

In [21]:
```
from sklearn.metrics import roc_curve
fpr, tpr, _ = roc_curve(ytest,pred)
auc = roc_auc_score(ytest, pred)
plt.figure(figsize=(12,8))
plt.plot(fpr,tpr,label="Validation AUC-ROC="+str(auc))
x = np.linspace(0, 1, 1000)
plt.plot(x, x, linestyle='-')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend(loc=4)
plt.show()
```



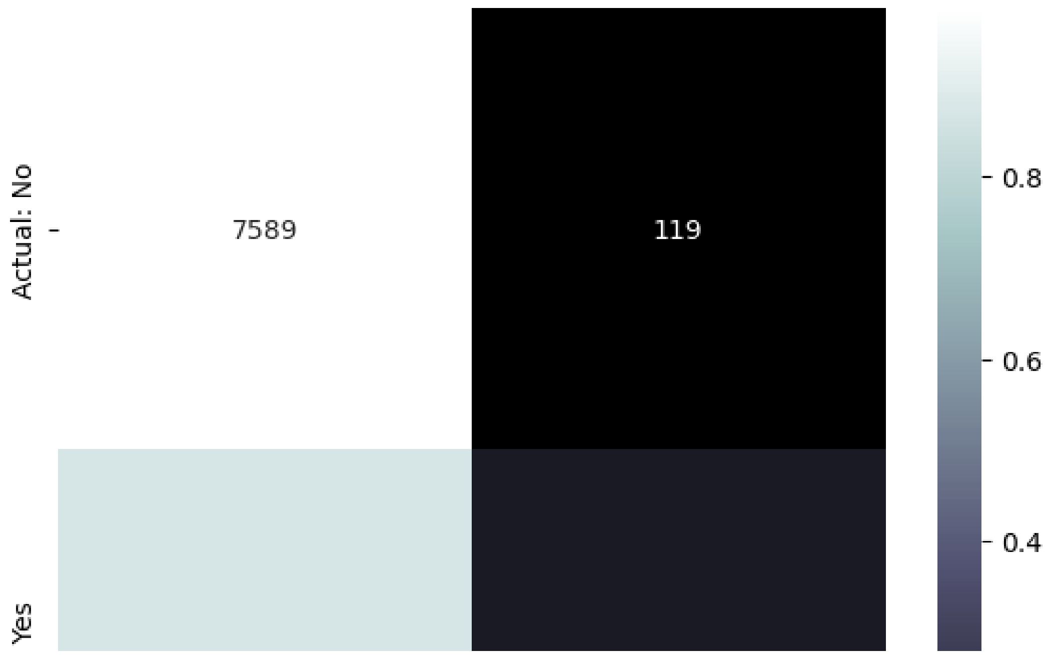
Confusion Matrix

In [23]:
```
pred_val = model.predict(xtest)
label_preds = pred_val

cm = confusion_matrix(ytest,label_preds)


def plot_confusion_matrix(cm, normalized=True, cmap='bone'):
    plt.figure(figsize=[7, 6])
    norm_cm = cm
    if normalized:
        norm_cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        sns.heatmap(norm_cm, annot=cm, fmt='g', xticklabels=['Predicted: N

plot_confusion_matrix(cm, ['No', 'Yes'])
```



In [24]:
```
recall_score(ytest,pred_val)
```

Out[24]: 0.13234455219623503

In [25]:

```
def cv_score(ml_model, rstate = 12, thres = 0.5, cols = df.columns):
    i = 1
    cv_scores = []
    df1 = df.copy()
    df1 = df[cols]

    # 5 Fold cross validation stratified on the basis of target
    kf = StratifiedKFold(n_splits=5,random_state=rstate,shuffle=True)
    for df_index,test_index in kf.split(df1,y_all):
        print('\n{} of kfold {}'.format(i,kf.n_splits))
        xtr,xvl = df1.loc[df_index],df1.loc[test_index]
        ytr,yvl = y_all.loc[df_index],y_all.loc[test_index]
baseline_scores = cv_score(LogisticRegression(), cols = baseline_cols)
```

```

1 of kfold 5

1 of kfold 5

1 of kfold 5

1 of kfold 5

1 of kfold 5
```

In [ ]:

```
```