8. Customer Churn Prediction and Handling Imbalance

December 22, 2024

[1]:
```python
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
```

[2]:
```python
df=pd.read_csv("D:\\docs\\telecomcustomerchurn.csv")
df.sample(5)
#here customer id is useless
```

[2]:
```
      customerID  gender  SeniorCitizen Partner Dependents  tenure  \
4470  8242-JSVBO    Male              0      No         No       7
5230  5887-IKKYO    Male              0     Yes        Yes      58
3725  0968-GSIKN  Female              0      No         No       1
3064  7855-DIWPO  Female              0      No         No      21
4978  4855-SNKMY  Female              0      No         No       1

     PhoneService MultipleLines InternetService OnlineSecurity  …  \
4470          Yes            No             DSL             No  …
5230          Yes           Yes     Fiber optic             No  …
3725          Yes            No     Fiber optic             No  …
3064          Yes            No     Fiber optic             No  …
4978          Yes            No             DSL             No  …

     DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
4470               No          No          No              No  Month-to-month
5230              Yes         Yes          No             Yes        Two year
3725               No          No          No              No  Month-to-month
3064               No          No          No              No  Month-to-month
4978               No          No          No              No  Month-to-month

     PaperlessBilling              PaymentMethod MonthlyCharges  TotalCharges  \
4470              Yes  Bank transfer (automatic)          44.65         322.5
5230               No  Bank transfer (automatic)          94.35       5563.65
3725              Yes               Mailed check          70.80          70.8
3064              Yes           Electronic check          68.65        1493.2
4978              Yes           Electronic check          44.10          44.1

     Churn
4470    No
```

```
5230     No
3725    Yes
3064     No
4978    Yes

[5 rows x 21 columns]
```

[3]:
```
df.drop('customerID',axis='columns',inplace=True)
df.dtypes
#here totalcharges is object but montly charges is float
```

[3]:
```
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

[4]:
```
df['TotalCharges'].values
```

[4]:
```
array(['29.85', '1889.5', '108.15', …, '346.45', '306.6', '6844.5'],
      dtype=object)
```

[5]:
```
pd.to_numeric(df.TotalCharges)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
File lib.pyx:2391, in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string " "

During handling of the above exception, another exception occurred:
```

```
ValueError                                Traceback (most recent call last)
Cell In[5], line 1
----> 1 pd.to_numeric(df.TotalCharges)
      2 #there are some spaces in between the values lets remove it

File
 ↪~\AppData\Local\Programs\Python\Python312\Lib\site-packages\pandas\core\tools\numeric.
 ↪py:232, in to_numeric(arg, errors, downcast, dtype_backend)
    230 coerce_numeric = errors not in ("ignore", "raise")
    231 try:
--> 232     values, new_mask =
 ↪lib.maybe_convert_numeric(  # type: ignore[call-overload]
    233         values,
    234         set(),
    235         coerce_numeric=coerce_numeric,
    236         convert_to_masked_nullable=dtype_backend is not lib.no_default
    237         or isinstance(values_dtype, StringDtype)
    238         and not values_dtype.storage == "pyarrow_numpy",
    239     )
    240 except (ValueError, TypeError):
    241     if errors == "raise":

File lib.pyx:2433, in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string " " at position 488
```

0.1 there are some spaces in between the values lets remove it

[6]:
```
pd.to_numeric(df.TotalCharges,errors='coerce').isnull()
```

[6]:
```
0       False
1       False
2       False
3       False
4       False
        …
7038    False
7039    False
7040    False
7041    False
7042    False
Name: TotalCharges, Length: 7043, dtype: bool
```

[7]:
```
df[pd.to_numeric(df.TotalCharges,errors='coerce').isnull()]
# let drop these rows
```

[7]:
```
      gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
488   Female              0     Yes        Yes       0           No
753     Male              0      No        Yes       0          Yes
936   Female              0     Yes        Yes       0          Yes
1082    Male              0     Yes        Yes       0          Yes
1340  Female              0     Yes        Yes       0           No
3331    Male              0     Yes        Yes       0          Yes
3826    Male              0     Yes        Yes       0          Yes
4380  Female              0     Yes        Yes       0          Yes
5218    Male              0     Yes        Yes       0          Yes
6670  Female              0     Yes        Yes       0          Yes
6754    Male              0      No        Yes       0          Yes

         MultipleLines InternetService       OnlineSecurity  \
488   No phone service             DSL                  Yes
753                 No              No  No internet service
936                 No             DSL                  Yes
1082               Yes              No  No internet service
1340  No phone service             DSL                  Yes
3331                No              No  No internet service
3826               Yes              No  No internet service
4380                No              No  No internet service
5218                No              No  No internet service
6670               Yes             DSL                   No
6754               Yes             DSL                  Yes

             OnlineBackup     DeviceProtection          TechSupport  \
488                    No                  Yes                  Yes
753   No internet service  No internet service  No internet service
936                   Yes                  Yes                   No
1082  No internet service  No internet service  No internet service
1340                  Yes                  Yes                  Yes
3331  No internet service  No internet service  No internet service
3826  No internet service  No internet service  No internet service
4380  No internet service  No internet service  No internet service
5218  No internet service  No internet service  No internet service
6670                  Yes                  Yes                  Yes
6754                  Yes                   No                  Yes

              StreamingTV      StreamingMovies  Contract PaperlessBilling  \
488                   Yes                   No  Two year              Yes
753   No internet service  No internet service  Two year               No
936                   Yes                  Yes  Two year               No
1082  No internet service  No internet service  Two year               No
1340                  Yes                   No  Two year               No
3331  No internet service  No internet service  Two year               No
3826  No internet service  No internet service  Two year               No
```

```
4380  No internet service  No internet service  Two year              No
5218  No internet service  No internet service  One year             Yes
6670                  Yes                   No  Two year              No
6754                   No                   No  Two year             Yes

                  PaymentMethod  MonthlyCharges TotalCharges Churn
488   Bank transfer (automatic)           52.55                 No
753                Mailed check           20.25                 No
936                Mailed check           80.85                 No
1082               Mailed check           25.75                 No
1340    Credit card (automatic)           56.05                 No
3331               Mailed check           19.85                 No
3826               Mailed check           25.35                 No
4380               Mailed check           20.00                 No
5218               Mailed check           19.70                 No
6670               Mailed check           73.35                 No
6754  Bank transfer (automatic)           61.90                 No
```

[8]:
```python
df.iloc[488]
```

[8]:
```
gender                                 Female
SeniorCitizen                               0
Partner                                   Yes
Dependents                                Yes
tenure                                      0
PhoneService                               No
MultipleLines                No phone service
InternetService                           DSL
OnlineSecurity                            Yes
OnlineBackup                               No
DeviceProtection                          Yes
TechSupport                               Yes
StreamingTV                               Yes
StreamingMovies                            No
Contract                             Two year
PaperlessBilling                          Yes
PaymentMethod       Bank transfer (automatic)
MonthlyCharges                          52.55
TotalCharges
Churn                                      No
Name: 488, dtype: object
```

[9]:
```python
df1=df[df.TotalCharges!=' ']
df1.shape
```

[9]:
```
(7032, 20)
```

[10]:
```
df1.dtypes
```

[10]:
```
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

[11]:
```
pd.to_numeric(df1.TotalCharges)
```

[11]:
```
0         29.85
1       1889.50
2        108.15
3       1840.75
4        151.65
          …
7038    1990.50
7039    7362.90
7040     346.45
7041     306.60
7042    6844.50
Name: TotalCharges, Length: 7032, dtype: float64
```

[12]:
```
df1.TotalCharges=pd.to_numeric(df1.TotalCharges)
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_17592\695980592.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
df1.TotalCharges=pd.to_numeric(df1.TotalCharges)
```

[13]:
```
df1.dtypes
#datatype is changes here
```

[13]:
```
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges        float64
Churn                object
dtype: object
```

1 tenure is how loyal a customer is

[14]:
```
df1[df1.Churn=='No']
```

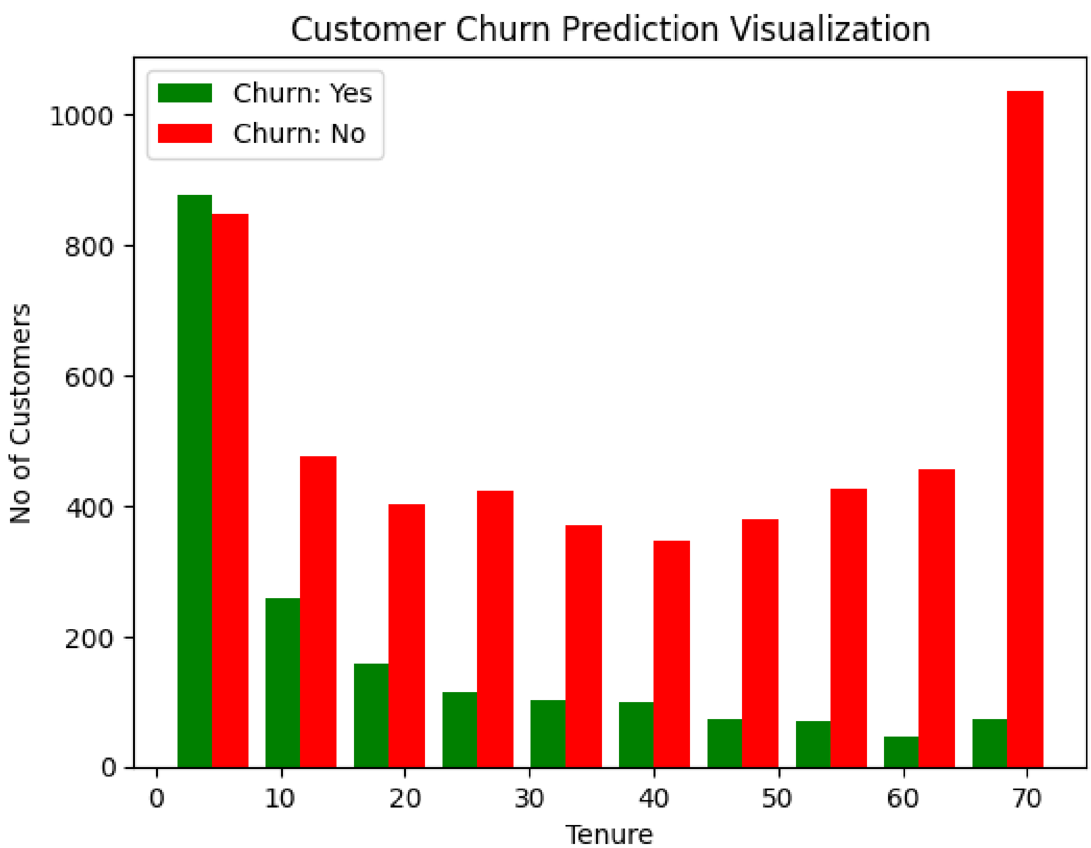
[14]:
```
      gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0     Female              0     Yes         No       1           No
1       Male              0      No         No      34          Yes
3       Male              0      No         No      45           No
6       Male              0      No        Yes      22          Yes
7     Female              0      No         No      10           No
…      …            …       …        …      …           …
7037  Female              0      No         No      72          Yes
7038    Male              0     Yes        Yes      24          Yes
7039  Female              0     Yes        Yes      72          Yes
7040  Female              0     Yes        Yes      11           No
7042    Male              0      No         No      66          Yes

         MultipleLines InternetService       OnlineSecurity  \
0     No phone service             DSL                   No
1                   No             DSL                  Yes
```

```
3     No phone service          DSL                 Yes
6                  Yes  Fiber optic                  No
7     No phone service          DSL                 Yes
…                    …            …                   …
7037                No           No  No internet service
7038               Yes          DSL                 Yes
7039               Yes  Fiber optic                  No
7040  No phone service          DSL                 Yes
7042                No  Fiber optic                 Yes

             OnlineBackup     DeviceProtection          TechSupport  \
0                     Yes                   No                   No
1                      No                  Yes                   No
3                      No                  Yes                  Yes
6                     Yes                   No                   No
7                      No                   No                   No
…                       …                    …                    …
7037  No internet service  No internet service  No internet service
7038                   No                  Yes                  Yes
7039                  Yes                  Yes                   No
7040                   No                   No                   No
7042                   No                  Yes                  Yes

              StreamingTV      StreamingMovies        Contract  \
0                      No                   No  Month-to-month
1                      No                   No        One year
3                      No                   No        One year
6                     Yes                   No  Month-to-month
7                      No                   No  Month-to-month
…                       …                    …               …
7037  No internet service  No internet service        Two year
7038                  Yes                  Yes        One year
7039                  Yes                  Yes        One year
7040                   No                   No  Month-to-month
7042                  Yes                  Yes        Two year

     PaperlessBilling              PaymentMethod  MonthlyCharges  \
0                 Yes           Electronic check           29.85
1                  No               Mailed check           56.95
3                  No  Bank transfer (automatic)           42.30
6                 Yes    Credit card (automatic)           89.10
7                  No               Mailed check           29.75
…                   …                          …               …
7037              Yes  Bank transfer (automatic)           21.15
7038              Yes               Mailed check           84.80
7039              Yes    Credit card (automatic)          103.20
7040              Yes           Electronic check           29.60
```

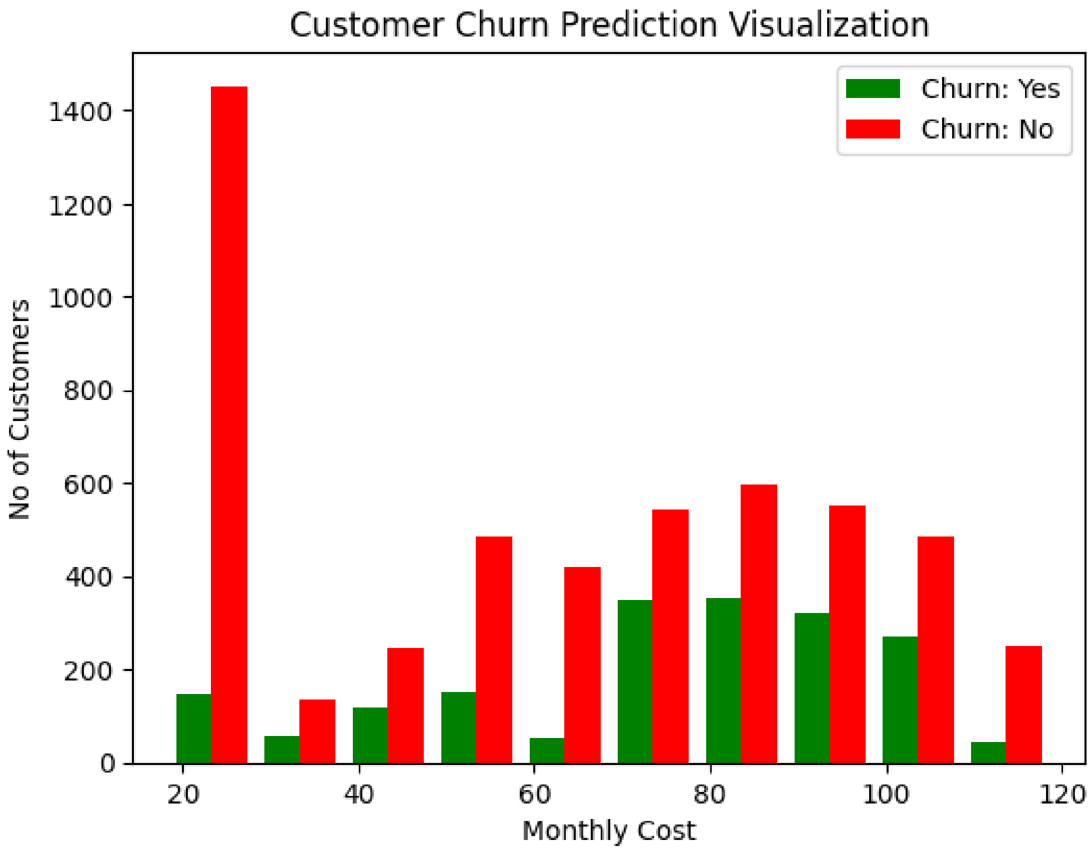
```
7042              Yes   Bank transfer (automatic)          105.65

      TotalCharges Churn
0            29.85    No
1          1889.50    No
3          1840.75    No
6          1949.40    No
7           301.90    No
…              …   …
7037       1419.40    No
7038       1990.50    No
7039       7362.90    No
7040        346.45    No
7042       6844.50    No

[5163 rows x 20 columns]
```

[15]:
```
tenure_churn_no=df1[df1.Churn=='No'].tenure
tenure_churn_yes=df1[df1.Churn=='Yes'].tenure
```

[16]:
```
plt.hist([tenure_churn_yes,tenure_churn_no],color=['green','red'],label=['Churn:
 ↪ Yes','Churn: No'])
plt.xlabel('Tenure')
plt.ylabel('No of Customers')
plt.title('Customer Churn Prediction Visualization')
plt.legend()
plt.show()
```



[17]:
```
monthlycost_churn_no=df1[df1.Churn=='No'].MonthlyCharges
monthlycost_churn_yes=df1[df1.Churn=='Yes'].MonthlyCharges
plt.
 ↪hist([monthlycost_churn_yes,monthlycost_churn_no],color=['green','red'],label=['Churn:
 ↪ Yes','Churn: No'])
plt.xlabel('Monthly Cost')
plt.ylabel('No of Customers')
plt.title('Customer Churn Prediction Visualization')
plt.legend()
plt.show()
```



[18]:
```
for column in df:
    print(column)
```

```
gender
SeniorCitizen
Partner
Dependents
tenure
PhoneService
MultipleLines
InternetService
OnlineSecurity
OnlineBackup
DeviceProtection
TechSupport
StreamingTV
StreamingMovies
Contract
PaperlessBilling
PaymentMethod
```

```
MonthlyCharges
TotalCharges
Churn
```

[19]:
```
for column in df:
    print(column,": ",df[column].unique())
```

```
gender :  ['Female' 'Male']
SeniorCitizen :  [0 1]
Partner :  ['Yes' 'No']
Dependents :  ['No' 'Yes']
tenure :  [ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17
27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26  0
 39]
PhoneService :  ['No' 'Yes']
MultipleLines :  ['No phone service' 'No' 'Yes']
InternetService :  ['DSL' 'Fiber optic' 'No']
OnlineSecurity :  ['No' 'Yes' 'No internet service']
OnlineBackup :  ['Yes' 'No' 'No internet service']
DeviceProtection :  ['No' 'Yes' 'No internet service']
TechSupport :  ['No' 'Yes' 'No internet service']
StreamingTV :  ['No' 'Yes' 'No internet service']
StreamingMovies :  ['No' 'Yes' 'No internet service']
Contract :  ['Month-to-month' 'One year' 'Two year']
PaperlessBilling :  ['Yes' 'No']
PaymentMethod :  ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
MonthlyCharges :  [29.85 56.95 53.85 … 63.1  44.2  78.7 ]
TotalCharges :  ['29.85' '1889.5' '108.15' … '346.45' '306.6' '6844.5']
Churn :  ['No' 'Yes']
```

[20]:
```
def print_unique_col_values(df):
    for column in df:
        if df[column].dtypes=='object':
            print(column,": ",df[column].unique())
```

[21]:
```
print_unique_col_values(df1)
```

```
gender :  ['Female' 'Male']
Partner :  ['Yes' 'No']
Dependents :  ['No' 'Yes']
PhoneService :  ['No' 'Yes']
MultipleLines :  ['No phone service' 'No' 'Yes']
InternetService :  ['DSL' 'Fiber optic' 'No']
OnlineSecurity :  ['No' 'Yes' 'No internet service']
OnlineBackup :  ['Yes' 'No' 'No internet service']
DeviceProtection :  ['No' 'Yes' 'No internet service']
```

```
TechSupport :  ['No' 'Yes' 'No internet service']
StreamingTV :  ['No' 'Yes' 'No internet service']
StreamingMovies :  ['No' 'Yes' 'No internet service']
Contract :  ['Month-to-month' 'One year' 'Two year']
PaperlessBilling :  ['Yes' 'No']
PaymentMethod :  ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
Churn :  ['No' 'Yes']
```

[22]:
```
df1.replace('No internet service','No',inplace=True)
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_17592\3939576099.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1.replace('No internet service','No',inplace=True)
```

[23]:
```
df1.replace('No phone service','No',inplace=True)
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_17592\628100714.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1.replace('No phone service','No',inplace=True)
```

[24]:
```
print_unique_col_values(df1)
```

```
gender :  ['Female' 'Male']
Partner :  ['Yes' 'No']
Dependents :  ['No' 'Yes']
PhoneService :  ['No' 'Yes']
MultipleLines :  ['No' 'Yes']
InternetService :  ['DSL' 'Fiber optic' 'No']
OnlineSecurity :  ['No' 'Yes']
OnlineBackup :  ['Yes' 'No']
DeviceProtection :  ['No' 'Yes']
TechSupport :  ['No' 'Yes']
StreamingTV :  ['No' 'Yes']
StreamingMovies :  ['No' 'Yes']
Contract :  ['Month-to-month' 'One year' 'Two year']
PaperlessBilling :  ['Yes' 'No']
PaymentMethod :  ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
Churn :  ['No' 'Yes']
```

[25]:
```
yes_no_columns=['Partner','Dependents','PhoneService','MultipleLines','OnlineSecurity','Online

for col in yes_no_columns:
    df1[col].replace({'Yes':1,'No':0},inplace=True)
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_17592\1255182669.py:4:
FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df1[col].replace({'Yes':1,'No':0},inplace=True)
C:\Users\admin\AppData\Local\Temp\ipykernel_17592\1255182669.py:4:
FutureWarning: Downcasting behavior in `replace` is deprecated and will be
removed in a future version. To retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
  df1[col].replace({'Yes':1,'No':0},inplace=True)
C:\Users\admin\AppData\Local\Temp\ipykernel_17592\1255182669.py:4:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1[col].replace({'Yes':1,'No':0},inplace=True)
```

[26]:
```
for col in df1:
    print(col,": ",df1[col].unique())
```

```
gender :  ['Female' 'Male']
SeniorCitizen :  [0 1]
Partner :  [1 0]
Dependents :  [0 1]
tenure :  [ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17
27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26 39]
PhoneService :  [0 1]
MultipleLines :  [0 1]
InternetService :  ['DSL' 'Fiber optic' 'No']
OnlineSecurity :  [0 1]
OnlineBackup :  [1 0]
```

```
DeviceProtection :  [0 1]
TechSupport :  [0 1]
StreamingTV :  [0 1]
StreamingMovies :  [0 1]
Contract :  ['Month-to-month' 'One year' 'Two year']
PaperlessBilling :  [1 0]
PaymentMethod :  ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
MonthlyCharges :  [29.85 56.95 53.85 … 63.1  44.2  78.7 ]
TotalCharges :  [  29.85 1889.5   108.15 …  346.45  306.6  6844.5 ]
Churn :  [0 1]
```

[27]:
```
df1['gender'].replace({'Female':1,'Male':0},inplace=True)
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_17592\698335744.py:1: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df1['gender'].replace({'Female':1,'Male':0},inplace=True)
C:\Users\admin\AppData\Local\Temp\ipykernel_17592\698335744.py:1: FutureWarning:
Downcasting behavior in `replace` is deprecated and will be removed in a future
version. To retain the old behavior, explicitly call
`result.infer_objects(copy=False)`. To opt-in to the future behavior, set
`pd.set_option('future.no_silent_downcasting', True)`
  df1['gender'].replace({'Female':1,'Male':0},inplace=True)
C:\Users\admin\AppData\Local\Temp\ipykernel_17592\698335744.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['gender'].replace({'Female':1,'Male':0},inplace=True)
```

[28]:
```
for col in df1:
    print(col,": ",df1[col].unique())
```

```
gender :  [1 0]
SeniorCitizen :  [0 1]
Partner :  [1 0]
Dependents :  [0 1]
tenure :  [ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17
```

```
27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26 39]
PhoneService :  [0 1]
MultipleLines :  [0 1]
InternetService :  ['DSL' 'Fiber optic' 'No']
OnlineSecurity :  [0 1]
OnlineBackup :  [1 0]
DeviceProtection :  [0 1]
TechSupport :  [0 1]
StreamingTV :  [0 1]
StreamingMovies :  [0 1]
Contract :  ['Month-to-month' 'One year' 'Two year']
PaperlessBilling :  [1 0]
PaymentMethod :  ['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
MonthlyCharges :  [29.85 56.95 53.85 … 63.1  44.2  78.7 ]
TotalCharges :  [  29.85 1889.5   108.15 …  346.45  306.6  6844.5 ]
Churn :  [0 1]
```

1.1 Now we will do one hot encoding for the columns internetservice, contract, paymentmethod etc

[29]:
```python
df2=pd.
 ↪get_dummies(data=df1,columns=['InternetService','Contract','PaymentMethod'],dtype=int)
df2.columns
```

[29]:
```
Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'OnlineSecurity', 'OnlineBackup',
       'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies',
       'PaperlessBilling', 'MonthlyCharges', 'TotalCharges', 'Churn',
       'InternetService_DSL', 'InternetService_Fiber optic',
       'InternetService_No', 'Contract_Month-to-month', 'Contract_One year',
       'Contract_Two year', 'PaymentMethod_Bank transfer (automatic)',
       'PaymentMethod_Credit card (automatic)',
       'PaymentMethod_Electronic check', 'PaymentMethod_Mailed check'],
      dtype='object')
```

[30]:
```python
df2
```

[30]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	\
0	1	0	1	0	1	0	
1	0	0	0	0	34	1	
2	0	0	0	0	2	1	
3	0	0	0	0	45	0	
4	1	0	0	0	2	1	
…	…	…	…	…	…	…	

```
7038       0              0        1            1      24              1
7039       1              0        1            1      72              1
7040       1              0        1            1      11              0
7041       0              1        1            0       4              1
7042       0              0        0            0      66              1

      MultipleLines  OnlineSecurity  OnlineBackup  DeviceProtection  …  \
0                 0               0             1                 0  …
1                 0               1             0                 1  …
2                 0               1             1                 0  …
3                 0               1             0                 1  …
4                 0               0             0                 0  …
…               …             …            …               …  …
7038              1               1             0                 1  …
7039              1               0             1                 1  …
7040              0               1             0                 0  …
7041              1               0             0                 0  …
7042              0               1             0                 1  …

      InternetService_DSL  InternetService_Fiber optic  InternetService_No  \
0                       1                            0                   0
1                       1                            0                   0
2                       1                            0                   0
3                       1                            0                   0
4                       0                            1                   0
…                     …                          …                 …
7038                    1                            0                   0
7039                    0                            1                   0
7040                    1                            0                   0
7041                    0                            1                   0
7042                    0                            1                   0

      Contract_Month-to-month  Contract_One year  Contract_Two year  \
0                           1                  0                  0
1                           0                  1                  0
2                           1                  0                  0
3                           0                  1                  0
4                           1                  0                  0
…                         …                …                …
7038                        0                  1                  0
7039                        0                  1                  0
7040                        1                  0                  0
7041                        1                  0                  0
7042                        0                  0                  1

      PaymentMethod_Bank transfer (automatic)  \
0                                           0
```

```
1                                          0
2                                          0
3                                          1
4                                          0
…                                      …
7038                                       0
7039                                       0
7040                                       0
7041                                       0
7042                                       1

      PaymentMethod_Credit card (automatic)  PaymentMethod_Electronic check  \
0                                        0                               1
1                                        0                               0
2                                        0                               0
3                                        0                               0
4                                        0                               1
…                                    …                             …
7038                                     0                               0
7039                                     1                               0
7040                                     0                               1
7041                                     0                               0
7042                                     0                               0

      PaymentMethod_Mailed check
0                             0
1                             1
2                             1
3                             0
4                             0
…                         …
7038                          1
7039                          0
7040                          0
7041                          1
7042                          0

[7032 rows x 27 columns]
```

[31]:
```
df2.dtypes
```

[31]:
```
gender                                    int64
SeniorCitizen                             int64
Partner                                   int64
Dependents                                int64
tenure                                    int64
PhoneService                              int64
```

```
MultipleLines                              int64
OnlineSecurity                             int64
OnlineBackup                               int64
DeviceProtection                           int64
TechSupport                                int64
StreamingTV                                int64
StreamingMovies                            int64
PaperlessBilling                           int64
MonthlyCharges                           float64
TotalCharges                             float64
Churn                                      int64
InternetService_DSL                        int32
InternetService_Fiber optic                int32
InternetService_No                         int32
Contract_Month-to-month                    int32
Contract_One year                          int32
Contract_Two year                          int32
PaymentMethod_Bank transfer (automatic)    int32
PaymentMethod_Credit card (automatic)      int32
PaymentMethod_Electronic check             int32
PaymentMethod_Mailed check                 int32
dtype: object
```

[32]:
```python
cols_to_scale=['tenure','MonthlyCharges','TotalCharges']
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
df2[cols_to_scale]=scaler.fit_transform(df2[cols_to_scale])
```

[33]:
```python
df2.sample(5)
```

[33]:
```
      gender  SeniorCitizen  Partner  Dependents    tenure  PhoneService  \
4623       0              1        1           0  0.957746             1
2980       1              0        0           0  1.000000             1
2867       1              0        0           0  0.323944             1
2966       1              1        0           0  0.183099             0
2068       0              0        0           0  0.647887             1

      MultipleLines  OnlineSecurity  OnlineBackup  DeviceProtection  …  \
4623              1               0             1                 1  …
2980              1               0             0                 0  …
2867              0               0             0                 0  …
2966              0               0             0                 0  …
2068              1               1             1                 0  …

      InternetService_DSL  InternetService_Fiber optic  InternetService_No  \
4623                    0                            1                   0
2980                    0                            0                   1
```

```
2867                       0                             0                   1
2966                       1                             0                   0
2068                       0                             1                   0

      Contract_Month-to-month  Contract_One year  Contract_Two year  \
4623                        1                  0                  0
2980                        0                  0                  1
2867                        0                  1                  0
2966                        1                  0                  0
2068                        1                  0                  0

      PaymentMethod_Bank transfer (automatic)  \
4623                                        1
2980                                        0
2867                                        0
2966                                        0
2068                                        0

      PaymentMethod_Credit card (automatic)  PaymentMethod_Electronic check  \
4623                                      0                               0
2980                                      1                               0
2867                                      1                               0
2966                                      0                               1
2068                                      0                               0

      PaymentMethod_Mailed check
4623                           0
2980                           0
2867                           0
2966                           0
2068                           1

[5 rows x 27 columns]
```

[34]:
```
for col in df2:
    print(col,": ",df2[col].unique())
```

```
gender :  [1 0]
SeniorCitizen :  [0 1]
Partner :  [1 0]
Dependents :  [0 1]
tenure :  [0.         0.46478873 0.01408451 0.61971831 0.09859155 0.29577465
 0.12676056 0.38028169 0.85915493 0.16901408 0.21126761 0.8028169
 0.67605634 0.33802817 0.95774648 0.71830986 0.98591549 0.28169014
 0.15492958 0.4084507  0.64788732 1.         0.22535211 0.36619718
 0.05633803 0.63380282 0.14084507 0.97183099 0.87323944 0.5915493
 0.1971831  0.83098592 0.23943662 0.91549296 0.11267606 0.02816901
 0.42253521 0.69014085 0.88732394 0.77464789 0.08450704 0.57746479
```

```
 0.47887324 0.66197183 0.3943662  0.90140845 0.52112676 0.94366197
 0.43661972 0.76056338 0.50704225 0.49295775 0.56338028 0.07042254
 0.04225352 0.45070423 0.92957746 0.30985915 0.78873239 0.84507042
 0.18309859 0.26760563 0.73239437 0.54929577 0.81690141 0.32394366
 0.6056338  0.25352113 0.74647887 0.70422535 0.35211268 0.53521127]
PhoneService :  [0 1]
MultipleLines :  [0 1]
OnlineSecurity :  [0 1]
OnlineBackup :  [1 0]
DeviceProtection :  [0 1]
TechSupport :  [0 1]
StreamingTV :  [0 1]
StreamingMovies :  [0 1]
PaperlessBilling :  [1 0]
MonthlyCharges :  [0.11542289 0.38507463 0.35422886 … 0.44626866 0.25820896
0.60149254]
TotalCharges :  [0.0012751  0.21586661 0.01031041 … 0.03780868 0.03321025
0.78764136]
Churn :  [0 1]
InternetService_DSL :  [1 0]
InternetService_Fiber optic :  [0 1]
InternetService_No :  [0 1]
Contract_Month-to-month :  [1 0]
Contract_One year :  [0 1]
Contract_Two year :  [0 1]
PaymentMethod_Bank transfer (automatic) :  [0 1]
PaymentMethod_Credit card (automatic) :  [0 1]
PaymentMethod_Electronic check :  [1 0]
PaymentMethod_Mailed check :  [0 1]
```

[35]:
```
x=df2.drop('Churn',axis='columns')
y=df2['Churn']
```

[36]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=5)
```

[37]:
```
x_train.shape
```

[37]:
```
(5625, 26)
```

[38]:
```
x_test.shape
```

[38]:
```
(1407, 26)
```

[39]:
```
len(x_train.columns)
```

[39]:
```
26
```

[40]:
```
import tensorflow as tf
from tensorflow import keras

model=keras.Sequential([
    keras.layers.Dense(20,input_shape=(26,),activation='relu'),
    keras.layers.Dense(1,activation='sigmoid')
])

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
model.fit(x_train,y_train,epochs=100)
```

```
Epoch 1/100

C:\Users\admin\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

176/176            1s 971us/step -
accuracy: 0.7339 - loss: 0.5356
Epoch 2/100
176/176            0s 539us/step -
accuracy: 0.7970 - loss: 0.4310
Epoch 3/100
176/176            0s 698us/step -
accuracy: 0.7909 - loss: 0.4229
Epoch 4/100
176/176            0s 573us/step -
accuracy: 0.8106 - loss: 0.4183
Epoch 5/100
176/176            0s 614us/step -
accuracy: 0.8099 - loss: 0.4091
Epoch 6/100
176/176            0s 579us/step -
accuracy: 0.8000 - loss: 0.4205
Epoch 7/100
176/176            0s 852us/step -
accuracy: 0.8070 - loss: 0.4118
Epoch 8/100
176/176            0s 1ms/step -
accuracy: 0.8063 - loss: 0.4136
Epoch 9/100
176/176            0s 570us/step -
accuracy: 0.8106 - loss: 0.4049
Epoch 10/100
176/176            0s 575us/step -
```

```
accuracy: 0.8029 - loss: 0.4120
Epoch 11/100
176/176              0s 839us/step -
accuracy: 0.8053 - loss: 0.4115
Epoch 12/100
176/176              0s 647us/step -
accuracy: 0.8106 - loss: 0.4068
Epoch 13/100
176/176              0s 548us/step -
accuracy: 0.8115 - loss: 0.4063
Epoch 14/100
176/176              0s 585us/step -
accuracy: 0.8073 - loss: 0.4022
Epoch 15/100
176/176              0s 644us/step -
accuracy: 0.8061 - loss: 0.4094
Epoch 16/100
176/176              0s 630us/step -
accuracy: 0.8025 - loss: 0.4131
Epoch 17/100
176/176              0s 617us/step -
accuracy: 0.8114 - loss: 0.3996
Epoch 18/100
176/176              0s 707us/step -
accuracy: 0.8108 - loss: 0.4006
Epoch 19/100
176/176              0s 549us/step -
accuracy: 0.8078 - loss: 0.4110
Epoch 20/100
176/176              0s 600us/step -
accuracy: 0.8122 - loss: 0.3972
Epoch 21/100
176/176              0s 604us/step -
accuracy: 0.8122 - loss: 0.3958
Epoch 22/100
176/176              0s 583us/step -
accuracy: 0.8066 - loss: 0.4029
Epoch 23/100
176/176              0s 635us/step -
accuracy: 0.8128 - loss: 0.3963
Epoch 24/100
176/176              0s 575us/step -
accuracy: 0.8105 - loss: 0.4041
Epoch 25/100
176/176              0s 919us/step -
accuracy: 0.8068 - loss: 0.4112
Epoch 26/100
176/176              0s 528us/step -
```

```
accuracy: 0.8185 - loss: 0.3940
Epoch 27/100
176/176              0s 606us/step -
accuracy: 0.8121 - loss: 0.4040
Epoch 28/100
176/176              0s 1ms/step -
accuracy: 0.8045 - loss: 0.4061
Epoch 29/100
176/176              0s 763us/step -
accuracy: 0.8114 - loss: 0.3953
Epoch 30/100
176/176              0s 569us/step -
accuracy: 0.8087 - loss: 0.4027
Epoch 31/100
176/176              0s 929us/step -
accuracy: 0.8174 - loss: 0.3899
Epoch 32/100
176/176              0s 552us/step -
accuracy: 0.8028 - loss: 0.4086
Epoch 33/100
176/176              0s 549us/step -
accuracy: 0.8098 - loss: 0.4012
Epoch 34/100
176/176              0s 556us/step -
accuracy: 0.8148 - loss: 0.3939
Epoch 35/100
176/176              0s 648us/step -
accuracy: 0.8194 - loss: 0.3886
Epoch 36/100
176/176              0s 615us/step -
accuracy: 0.8162 - loss: 0.3900
Epoch 37/100
176/176              0s 575us/step -
accuracy: 0.8186 - loss: 0.3924
Epoch 38/100
176/176              0s 556us/step -
accuracy: 0.8087 - loss: 0.4030
Epoch 39/100
176/176              0s 589us/step -
accuracy: 0.8125 - loss: 0.3952
Epoch 40/100
176/176              0s 560us/step -
accuracy: 0.8156 - loss: 0.3924
Epoch 41/100
176/176              0s 598us/step -
accuracy: 0.8117 - loss: 0.3844
Epoch 42/100
176/176              0s 625us/step -
```

```
accuracy: 0.8119 - loss: 0.3942
Epoch 43/100
176/176              0s 627us/step -
accuracy: 0.8290 - loss: 0.3709
Epoch 44/100
176/176              0s 652us/step -
accuracy: 0.8201 - loss: 0.3816
Epoch 45/100
176/176              0s 520us/step -
accuracy: 0.8130 - loss: 0.3975
Epoch 46/100
176/176              0s 621us/step -
accuracy: 0.8200 - loss: 0.3817
Epoch 47/100
176/176              0s 568us/step -
accuracy: 0.8196 - loss: 0.3830
Epoch 48/100
176/176              0s 642us/step -
accuracy: 0.8163 - loss: 0.3919
Epoch 49/100
176/176              0s 620us/step -
accuracy: 0.8215 - loss: 0.3754
Epoch 50/100
176/176              0s 731us/step -
accuracy: 0.8277 - loss: 0.3708
Epoch 51/100
176/176              0s 634us/step -
accuracy: 0.8137 - loss: 0.3953
Epoch 52/100
176/176              0s 690us/step -
accuracy: 0.8177 - loss: 0.3886
Epoch 53/100
176/176              0s 620us/step -
accuracy: 0.8161 - loss: 0.3776
Epoch 54/100
176/176              0s 646us/step -
accuracy: 0.8084 - loss: 0.4029
Epoch 55/100
176/176              0s 629us/step -
accuracy: 0.8155 - loss: 0.3937
Epoch 56/100
176/176              0s 618us/step -
accuracy: 0.8132 - loss: 0.3907
Epoch 57/100
176/176              0s 614us/step -
accuracy: 0.8153 - loss: 0.3808
Epoch 58/100
176/176              0s 768us/step -
```

```
accuracy: 0.8220 - loss: 0.3785
Epoch 59/100
176/176              0s 558us/step -
accuracy: 0.8235 - loss: 0.3820
Epoch 60/100
176/176              0s 563us/step -
accuracy: 0.8166 - loss: 0.3926
Epoch 61/100
176/176              0s 604us/step -
accuracy: 0.8198 - loss: 0.3847
Epoch 62/100
176/176              0s 579us/step -
accuracy: 0.8213 - loss: 0.3772
Epoch 63/100
176/176              0s 623us/step -
accuracy: 0.8266 - loss: 0.3781
Epoch 64/100
176/176              0s 664us/step -
accuracy: 0.8258 - loss: 0.3798
Epoch 65/100
176/176              0s 611us/step -
accuracy: 0.8308 - loss: 0.3715
Epoch 66/100
176/176              0s 612us/step -
accuracy: 0.8199 - loss: 0.3784
Epoch 67/100
176/176              0s 639us/step -
accuracy: 0.8158 - loss: 0.3816
Epoch 68/100
176/176              0s 630us/step -
accuracy: 0.8159 - loss: 0.3840
Epoch 69/100
176/176              0s 617us/step -
accuracy: 0.8245 - loss: 0.3748
Epoch 70/100
176/176              0s 624us/step -
accuracy: 0.8058 - loss: 0.3977
Epoch 71/100
176/176              0s 624us/step -
accuracy: 0.8163 - loss: 0.3883
Epoch 72/100
176/176              0s 618us/step -
accuracy: 0.8153 - loss: 0.3898
Epoch 73/100
176/176              0s 666us/step -
accuracy: 0.8136 - loss: 0.3874
Epoch 74/100
176/176              0s 939us/step -
```

```
accuracy: 0.8201 - loss: 0.3789
Epoch 75/100
176/176              0s 614us/step -
accuracy: 0.8185 - loss: 0.3799
Epoch 76/100
176/176              0s 634us/step -
accuracy: 0.8183 - loss: 0.3889
Epoch 77/100
176/176              0s 608us/step -
accuracy: 0.8231 - loss: 0.3813
Epoch 78/100
176/176              0s 650us/step -
accuracy: 0.8140 - loss: 0.3849
Epoch 79/100
176/176              0s 671us/step -
accuracy: 0.8176 - loss: 0.3792
Epoch 80/100
176/176              0s 815us/step -
accuracy: 0.8127 - loss: 0.3869
Epoch 81/100
176/176              0s 649us/step -
accuracy: 0.8248 - loss: 0.3720
Epoch 82/100
176/176              0s 626us/step -
accuracy: 0.8211 - loss: 0.3739
Epoch 83/100
176/176              0s 576us/step -
accuracy: 0.8264 - loss: 0.3744
Epoch 84/100
176/176              0s 629us/step -
accuracy: 0.8186 - loss: 0.3842
Epoch 85/100
176/176              0s 651us/step -
accuracy: 0.8210 - loss: 0.3759
Epoch 86/100
176/176              0s 794us/step -
accuracy: 0.8345 - loss: 0.3609
Epoch 87/100
176/176              0s 602us/step -
accuracy: 0.8278 - loss: 0.3652
Epoch 88/100
176/176              0s 637us/step -
accuracy: 0.8236 - loss: 0.3820
Epoch 89/100
176/176              0s 685us/step -
accuracy: 0.8225 - loss: 0.3836
Epoch 90/100
176/176              0s 666us/step -
```

```
accuracy: 0.8170 - loss: 0.3820
Epoch 91/100
176/176              0s 649us/step -
accuracy: 0.8243 - loss: 0.3672
Epoch 92/100
176/176              0s 663us/step -
accuracy: 0.8144 - loss: 0.3836
Epoch 93/100
176/176              0s 693us/step -
accuracy: 0.8187 - loss: 0.3781
Epoch 94/100
176/176              0s 648us/step -
accuracy: 0.8167 - loss: 0.3777
Epoch 95/100
176/176              0s 645us/step -
accuracy: 0.8213 - loss: 0.3707
Epoch 96/100
176/176              0s 639us/step -
accuracy: 0.8212 - loss: 0.3707
Epoch 97/100
176/176              0s 672us/step -
accuracy: 0.8170 - loss: 0.3771
Epoch 98/100
176/176              0s 833us/step -
accuracy: 0.8316 - loss: 0.3663
Epoch 99/100
176/176              0s 653us/step -
accuracy: 0.8209 - loss: 0.3767
Epoch 100/100
176/176              0s 657us/step -
accuracy: 0.8358 - loss: 0.3666
```

[40]:
```
<keras.src.callbacks.history.History at 0x1f5c0949cd0>
```

[41]:
```
model.evaluate(x_test,y_test)
```

```
44/44              0s 649us/step -
accuracy: 0.7886 - loss: 0.4486
```

[41]:
```
[0.46417662501335144, 0.7761194109916687]
```

[42]:
```
yp=model.predict(x_test)
yp[:5]
```

```
44/44              0s 1ms/step
```

[42]:
```
array([[0.21456003],
       [0.53348875],
       [0.01222272],
```

```
        [0.77699953],
        [0.55470014]], dtype=float32)
```

[43]:
```
y_pred=[]
for element in yp:
    if element>0.5:
        y_pred.append(1)
    else:
        y_pred.append(0)
```

[44]:
```
y_test[:10]
```

[44]:
```
2660    0
744     0
5579    1
64      1
3287    1
816     1
2670    0
5920    0
1023    0
6087    0
Name: Churn, dtype: int64
```

[45]:
```
y_pred[:10]
```

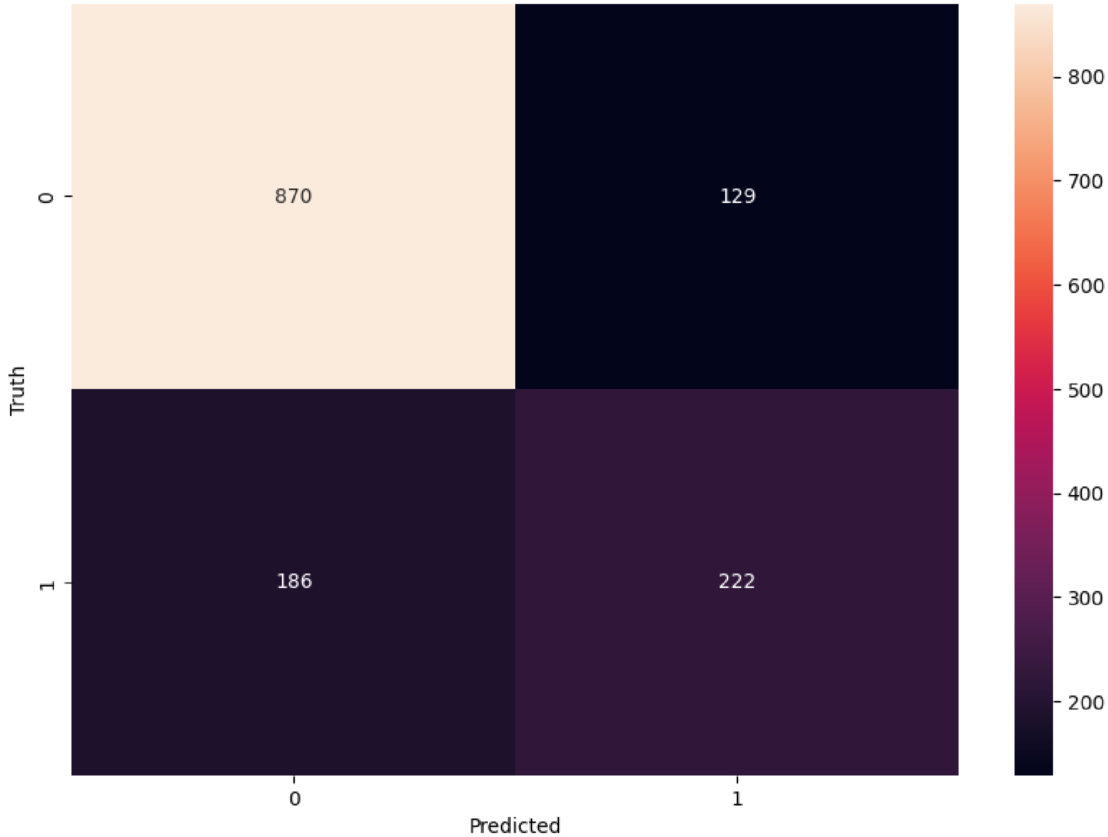
[45]:
```
[0, 1, 0, 1, 1, 1, 0, 0, 0, 0]
```

[46]:
```
from sklearn.metrics import confusion_matrix,classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.87      0.85       999
           1       0.63      0.54      0.58       408

    accuracy                           0.78      1407
   macro avg       0.73      0.71      0.72      1407
weighted avg       0.77      0.78      0.77      1407
```

[47]:
```
import seaborn as sn
cm=tf.math.confusion_matrix(labels=y_test,predictions=y_pred)

plt.figure(figsize=(10,7))
sn.heatmap(cm,annot=True,fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

[47]: Text(95.72222222222221, 0.5, 'Truth')




```
[52]: def ANN(x_train,y_train,x_test,y_test,loss,weights):
    model=keras.Sequential([
        keras.layers.Dense(26,input_dim=26,activation='relu'),
        keras.layers.Dense(15,activation='relu'),
        keras.layers.Dense(1,activation='sigmoid')
    ])
    model.compile(optimizer='adam',loss=loss,metrics=['accuracy'])

    if weights==-1:
        model.fit(x_train,y_train,epochs=100)
    else:
        model.fit(x_train,y_train,epochs=100,class_weight=weights)
    print(model.evaluate(x_test,y_test))
    y_preds=model.predict(x_test)
    y_preds=np.round(y_preds)
    print("Classification Report: \n",classification_report(y_test,y_preds))
    return y_preds
```

[53]:
```
y_preds=ANN(x_train,y_train,x_test,y_test,'binary_crossentropy',-1)
```

```
Epoch 1/100
```

```
C:\Users\admin\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
176/176             1s 710us/step -
accuracy: 0.6344 - loss: 0.6013
Epoch 2/100
176/176             0s 666us/step -
accuracy: 0.7898 - loss: 0.4390
Epoch 3/100
176/176             0s 663us/step -
accuracy: 0.7985 - loss: 0.4139
Epoch 4/100
176/176             0s 661us/step -
accuracy: 0.7953 - loss: 0.4224
Epoch 5/100
176/176             0s 705us/step -
accuracy: 0.8052 - loss: 0.4180
Epoch 6/100
176/176             0s 711us/step -
accuracy: 0.8128 - loss: 0.4102
Epoch 7/100
176/176             0s 698us/step -
accuracy: 0.8190 - loss: 0.3979
Epoch 8/100
176/176             0s 706us/step -
accuracy: 0.8099 - loss: 0.4087
Epoch 9/100
176/176             0s 691us/step -
accuracy: 0.8115 - loss: 0.3988
Epoch 10/100
176/176             0s 676us/step -
accuracy: 0.8104 - loss: 0.4103
Epoch 11/100
176/176             0s 658us/step -
accuracy: 0.8152 - loss: 0.3976
Epoch 12/100
176/176             0s 693us/step -
accuracy: 0.8162 - loss: 0.3947
Epoch 13/100
176/176             0s 672us/step -
accuracy: 0.8182 - loss: 0.3952
Epoch 14/100
```

```
176/176              0s 696us/step -
accuracy: 0.8082 - loss: 0.3984
Epoch 15/100
176/176              0s 689us/step -
accuracy: 0.8097 - loss: 0.4085
Epoch 16/100
176/176              0s 639us/step -
accuracy: 0.8160 - loss: 0.3923
Epoch 17/100
176/176              0s 688us/step -
accuracy: 0.8184 - loss: 0.3896
Epoch 18/100
176/176              0s 689us/step -
accuracy: 0.8131 - loss: 0.3909
Epoch 19/100
176/176              0s 848us/step -
accuracy: 0.8236 - loss: 0.3878
Epoch 20/100
176/176              0s 678us/step -
accuracy: 0.8088 - loss: 0.4046
Epoch 21/100
176/176              0s 681us/step -
accuracy: 0.8184 - loss: 0.3902
Epoch 22/100
176/176              0s 673us/step -
accuracy: 0.8190 - loss: 0.3831
Epoch 23/100
176/176              0s 693us/step -
accuracy: 0.8176 - loss: 0.3790
Epoch 24/100
176/176              0s 665us/step -
accuracy: 0.8198 - loss: 0.3889
Epoch 25/100
176/176              0s 675us/step -
accuracy: 0.8308 - loss: 0.3651
Epoch 26/100
176/176              0s 660us/step -
accuracy: 0.8142 - loss: 0.3934
Epoch 27/100
176/176              0s 653us/step -
accuracy: 0.8125 - loss: 0.3915
Epoch 28/100
176/176              0s 671us/step -
accuracy: 0.8131 - loss: 0.3874
Epoch 29/100
176/176              0s 843us/step -
accuracy: 0.8172 - loss: 0.3912
Epoch 30/100
```

```
176/176              0s 660us/step -
accuracy: 0.8159 - loss: 0.3801
Epoch 31/100
176/176              0s 667us/step -
accuracy: 0.8229 - loss: 0.3734
Epoch 32/100
176/176              0s 677us/step -
accuracy: 0.8234 - loss: 0.3765
Epoch 33/100
176/176              0s 683us/step -
accuracy: 0.8204 - loss: 0.3768
Epoch 34/100
176/176              0s 672us/step -
accuracy: 0.8213 - loss: 0.3741
Epoch 35/100
176/176              0s 655us/step -
accuracy: 0.8191 - loss: 0.3767
Epoch 36/100
176/176              0s 671us/step -
accuracy: 0.8206 - loss: 0.3763
Epoch 37/100
176/176              0s 654us/step -
accuracy: 0.8241 - loss: 0.3769
Epoch 38/100
176/176              0s 690us/step -
accuracy: 0.8241 - loss: 0.3788
Epoch 39/100
176/176              0s 655us/step -
accuracy: 0.8266 - loss: 0.3654
Epoch 40/100
176/176              0s 843us/step -
accuracy: 0.8160 - loss: 0.3843
Epoch 41/100
176/176              0s 657us/step -
accuracy: 0.8274 - loss: 0.3673
Epoch 42/100
176/176              0s 680us/step -
accuracy: 0.8162 - loss: 0.3794
Epoch 43/100
176/176              0s 657us/step -
accuracy: 0.8302 - loss: 0.3655
Epoch 44/100
176/176              0s 667us/step -
accuracy: 0.8216 - loss: 0.3703
Epoch 45/100
176/176              0s 671us/step -
accuracy: 0.8238 - loss: 0.3726
Epoch 46/100
```

```
176/176              0s 658us/step -
accuracy: 0.8194 - loss: 0.3770
Epoch 47/100
176/176              0s 667us/step -
accuracy: 0.8246 - loss: 0.3685
Epoch 48/100
176/176              0s 654us/step -
accuracy: 0.8248 - loss: 0.3703
Epoch 49/100
176/176              0s 726us/step -
accuracy: 0.8299 - loss: 0.3619
Epoch 50/100
176/176              0s 676us/step -
accuracy: 0.8177 - loss: 0.3751
Epoch 51/100
176/176              0s 684us/step -
accuracy: 0.8278 - loss: 0.3652
Epoch 52/100
176/176              0s 636us/step -
accuracy: 0.8229 - loss: 0.3665
Epoch 53/100
176/176              0s 642us/step -
accuracy: 0.8241 - loss: 0.3721
Epoch 54/100
176/176              0s 653us/step -
accuracy: 0.8285 - loss: 0.3674
Epoch 55/100
176/176              0s 642us/step -
accuracy: 0.8372 - loss: 0.3527
Epoch 56/100
176/176              0s 629us/step -
accuracy: 0.8288 - loss: 0.3680
Epoch 57/100
176/176              0s 640us/step -
accuracy: 0.8382 - loss: 0.3559
Epoch 58/100
176/176              0s 653us/step -
accuracy: 0.8201 - loss: 0.3619
Epoch 59/100
176/176              0s 687us/step -
accuracy: 0.8248 - loss: 0.3638
Epoch 60/100
176/176              0s 787us/step -
accuracy: 0.8246 - loss: 0.3668
Epoch 61/100
176/176              0s 719us/step -
accuracy: 0.8324 - loss: 0.3574
Epoch 62/100
```

```
176/176              0s 781us/step -
accuracy: 0.8347 - loss: 0.3582
Epoch 63/100
176/176              0s 653us/step -
accuracy: 0.8266 - loss: 0.3626
Epoch 64/100
176/176              0s 708us/step -
accuracy: 0.8250 - loss: 0.3682
Epoch 65/100
176/176              0s 640us/step -
accuracy: 0.8277 - loss: 0.3593
Epoch 66/100
176/176              0s 648us/step -
accuracy: 0.8321 - loss: 0.3563
Epoch 67/100
176/176              0s 648us/step -
accuracy: 0.8200 - loss: 0.3662
Epoch 68/100
176/176              0s 656us/step -
accuracy: 0.8282 - loss: 0.3622
Epoch 69/100
176/176              0s 650us/step -
accuracy: 0.8184 - loss: 0.3673
Epoch 70/100
176/176              0s 836us/step -
accuracy: 0.8280 - loss: 0.3564
Epoch 71/100
176/176              0s 680us/step -
accuracy: 0.8289 - loss: 0.3507
Epoch 72/100
176/176              0s 652us/step -
accuracy: 0.8197 - loss: 0.3645
Epoch 73/100
176/176              0s 663us/step -
accuracy: 0.8226 - loss: 0.3581
Epoch 74/100
176/176              0s 646us/step -
accuracy: 0.8340 - loss: 0.3534
Epoch 75/100
176/176              0s 648us/step -
accuracy: 0.8268 - loss: 0.3548
Epoch 76/100
176/176              0s 642us/step -
accuracy: 0.8299 - loss: 0.3590
Epoch 77/100
176/176              0s 773us/step -
accuracy: 0.8359 - loss: 0.3516
Epoch 78/100
```

```
176/176              0s 659us/step -
accuracy: 0.8299 - loss: 0.3503
Epoch 79/100
176/176              0s 661us/step -
accuracy: 0.8391 - loss: 0.3494
Epoch 80/100
176/176              0s 651us/step -
accuracy: 0.8382 - loss: 0.3377
Epoch 81/100
176/176              0s 651us/step -
accuracy: 0.8374 - loss: 0.3470
Epoch 82/100
176/176              0s 642us/step -
accuracy: 0.8322 - loss: 0.3586
Epoch 83/100
176/176              0s 659us/step -
accuracy: 0.8371 - loss: 0.3433
Epoch 84/100
176/176              0s 680us/step -
accuracy: 0.8322 - loss: 0.3426
Epoch 85/100
176/176              0s 648us/step -
accuracy: 0.8398 - loss: 0.3419
Epoch 86/100
176/176              0s 667us/step -
accuracy: 0.8344 - loss: 0.3484
Epoch 87/100
176/176              0s 633us/step -
accuracy: 0.8376 - loss: 0.3426
Epoch 88/100
176/176              0s 653us/step -
accuracy: 0.8376 - loss: 0.3549
Epoch 89/100
176/176              0s 837us/step -
accuracy: 0.8281 - loss: 0.3512
Epoch 90/100
176/176              0s 672us/step -
accuracy: 0.8380 - loss: 0.3400
Epoch 91/100
176/176              0s 659us/step -
accuracy: 0.8293 - loss: 0.3466
Epoch 92/100
176/176              0s 655us/step -
accuracy: 0.8420 - loss: 0.3393
Epoch 93/100
176/176              0s 648us/step -
accuracy: 0.8323 - loss: 0.3476
Epoch 94/100
```

```
176/176              0s 633us/step -
accuracy: 0.8280 - loss: 0.3575
Epoch 95/100
176/176              0s 820us/step -
accuracy: 0.8452 - loss: 0.3321
Epoch 96/100
176/176              0s 639us/step -
accuracy: 0.8341 - loss: 0.3506
Epoch 97/100
176/176              0s 682us/step -
accuracy: 0.8431 - loss: 0.3388
Epoch 98/100
176/176              0s 703us/step -
accuracy: 0.8408 - loss: 0.3367
Epoch 99/100
176/176              0s 645us/step -
accuracy: 0.8376 - loss: 0.3416
Epoch 100/100
176/176              0s 658us/step -
accuracy: 0.8377 - loss: 0.3459
44/44              0s 481us/step -
accuracy: 0.7759 - loss: 0.4851
[0.4905689060688019, 0.7690120935440063]
44/44              0s 2ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.86      0.84       999
           1       0.61      0.55      0.58       408

    accuracy                           0.77      1407
   macro avg       0.72      0.70      0.71      1407
weighted avg       0.76      0.77      0.77      1407
```

2 Method 1: Undersampling

[54]:
```python
count_class_0,count_class_1=df1.Churn.value_counts()

df_class_0=df2[df2['Churn']==0]
df_class_1=df2[df2['Churn']==1]
```

[62]:
```python
df_class_0['Churn'],df_class_1['Churn']
```

[62]:
```
(0       0
 1       0
 3       0
```

```
6       0
7       0
        ..
7037    0
7038    0
7039    0
7040    0
7042    0
Name: Churn, Length: 5163, dtype: int64,
2       1
4       1
5       1
8       1
13      1
        ..
7021    1
7026    1
7032    1
7034    1
7041    1
Name: Churn, Length: 1869, dtype: int64)
```

[55]:
```
df_class_0.shape
```

[55]: (5163, 27)

[56]:
```
df_class_1.shape
```

[56]: (1869, 27)

[68]:
```
df_class_0_under=df_class_0.sample(count_class_1)
df_test_under=pd.concat([df_class_0_under,df_class_1],axis='rows')
df_test_under.shape
```

[68]: (3738, 27)

[69]:
```
print("Random under-sampling: ")
print(df_test_under.Churn.value_counts())
```

```
Random under-sampling:
Churn
0    1869
1    1869
Name: count, dtype: int64
```

[70]:
```
df2.Churn.value_counts()
```

[70]:
```
Churn
0    5163
1    1869
Name: count, dtype: int64
```

[71]:
```
x=df_test_under.drop('Churn',axis='columns')
y=df_test_under['Churn']
```

[72]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
↪2,random_state=15,stratify=y)
```

[74]:
```
y_train.value_counts() #it is equal because of strartify=y
```

[74]:
```
Churn
0    1495
1    1495
Name: count, dtype: int64
```

[75]:
```
y_preds=ANN(x_train,y_train,x_test,y_test,'binary_crossentropy',-1)
```

```
C:\Users\admin\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
Epoch 1/100
94/94                 3s 2ms/step -
accuracy: 0.5963 - loss: 0.6593
Epoch 2/100
94/94                 0s 2ms/step -
accuracy: 0.7706 - loss: 0.5104
Epoch 3/100
94/94                 0s 2ms/step -
accuracy: 0.7707 - loss: 0.4960
Epoch 4/100
94/94                 0s 2ms/step -
accuracy: 0.7743 - loss: 0.4840
Epoch 5/100
94/94                 0s 2ms/step -
accuracy: 0.7795 - loss: 0.4635
Epoch 6/100
94/94                 0s 2ms/step -
accuracy: 0.7719 - loss: 0.4721
Epoch 7/100
94/94                 0s 2ms/step -
accuracy: 0.7964 - loss: 0.4605
Epoch 8/100
```

```
94/94              0s 3ms/step -
accuracy: 0.7869 - loss: 0.4605
Epoch 9/100
94/94              0s 2ms/step -
accuracy: 0.7904 - loss: 0.4528
Epoch 10/100
94/94              0s 2ms/step -
accuracy: 0.7794 - loss: 0.4715
Epoch 11/100
94/94              0s 2ms/step -
accuracy: 0.7773 - loss: 0.4676
Epoch 12/100
94/94              0s 3ms/step -
accuracy: 0.7819 - loss: 0.4553
Epoch 13/100
94/94              0s 2ms/step -
accuracy: 0.7797 - loss: 0.4669
Epoch 14/100
94/94              0s 2ms/step -
accuracy: 0.7979 - loss: 0.4461
Epoch 15/100
94/94              0s 2ms/step -
accuracy: 0.7991 - loss: 0.4349
Epoch 16/100
94/94              0s 2ms/step -
accuracy: 0.7862 - loss: 0.4461
Epoch 17/100
94/94              0s 3ms/step -
accuracy: 0.8023 - loss: 0.4332
Epoch 18/100
94/94              0s 2ms/step -
accuracy: 0.8028 - loss: 0.4433
Epoch 19/100
94/94              0s 2ms/step -
accuracy: 0.8086 - loss: 0.4332
Epoch 20/100
94/94              0s 2ms/step -
accuracy: 0.7969 - loss: 0.4385
Epoch 21/100
94/94              0s 2ms/step -
accuracy: 0.7993 - loss: 0.4447
Epoch 22/100
94/94              0s 2ms/step -
accuracy: 0.7906 - loss: 0.4532
Epoch 23/100
94/94              0s 2ms/step -
accuracy: 0.7945 - loss: 0.4478
Epoch 24/100
```

```
94/94             0s 2ms/step -
accuracy: 0.7951 - loss: 0.4259
Epoch 25/100
94/94             0s 2ms/step -
accuracy: 0.7936 - loss: 0.4451
Epoch 26/100
94/94             0s 2ms/step -
accuracy: 0.8016 - loss: 0.4338
Epoch 27/100
94/94             0s 2ms/step -
accuracy: 0.8061 - loss: 0.4301
Epoch 28/100
94/94             0s 2ms/step -
accuracy: 0.8013 - loss: 0.4331
Epoch 29/100
94/94             0s 2ms/step -
accuracy: 0.7950 - loss: 0.4366
Epoch 30/100
94/94             0s 2ms/step -
accuracy: 0.8053 - loss: 0.4285
Epoch 31/100
94/94             0s 2ms/step -
accuracy: 0.8014 - loss: 0.4378
Epoch 32/100
94/94             0s 2ms/step -
accuracy: 0.7947 - loss: 0.4270
Epoch 33/100
94/94             0s 2ms/step -
accuracy: 0.8110 - loss: 0.4166
Epoch 34/100
94/94             0s 2ms/step -
accuracy: 0.8035 - loss: 0.4266
Epoch 35/100
94/94             0s 2ms/step -
accuracy: 0.8030 - loss: 0.4267
Epoch 36/100
94/94             0s 2ms/step -
accuracy: 0.8091 - loss: 0.4122
Epoch 37/100
94/94             0s 2ms/step -
accuracy: 0.8090 - loss: 0.4231
Epoch 38/100
94/94             0s 2ms/step -
accuracy: 0.8029 - loss: 0.4158
Epoch 39/100
94/94             0s 2ms/step -
accuracy: 0.8127 - loss: 0.4106
Epoch 40/100
```

```
94/94              0s 2ms/step -
accuracy: 0.8048 - loss: 0.4206
Epoch 41/100
94/94              0s 2ms/step -
accuracy: 0.7856 - loss: 0.4404
Epoch 42/100
94/94              0s 2ms/step -
accuracy: 0.8103 - loss: 0.4124
Epoch 43/100
94/94              0s 2ms/step -
accuracy: 0.8200 - loss: 0.4050
Epoch 44/100
94/94              0s 2ms/step -
accuracy: 0.7985 - loss: 0.4292
Epoch 45/100
94/94              0s 2ms/step -
accuracy: 0.8118 - loss: 0.4047
Epoch 46/100
94/94              0s 2ms/step -
accuracy: 0.8222 - loss: 0.4055
Epoch 47/100
94/94              0s 2ms/step -
accuracy: 0.8172 - loss: 0.4032
Epoch 48/100
94/94              0s 3ms/step -
accuracy: 0.8205 - loss: 0.3987
Epoch 49/100
94/94              0s 2ms/step -
accuracy: 0.8023 - loss: 0.4221
Epoch 50/100
94/94              0s 2ms/step -
accuracy: 0.8134 - loss: 0.4076
Epoch 51/100
94/94              0s 2ms/step -
accuracy: 0.8195 - loss: 0.4029
Epoch 52/100
94/94              0s 2ms/step -
accuracy: 0.8155 - loss: 0.4051
Epoch 53/100
94/94              0s 2ms/step -
accuracy: 0.8174 - loss: 0.4014
Epoch 54/100
94/94              0s 2ms/step -
accuracy: 0.8244 - loss: 0.3984
Epoch 55/100
94/94              0s 2ms/step -
accuracy: 0.8167 - loss: 0.4000
Epoch 56/100
```

```
94/94             0s 2ms/step -
accuracy: 0.8290 - loss: 0.3921
Epoch 57/100
94/94             0s 2ms/step -
accuracy: 0.8189 - loss: 0.4018
Epoch 58/100
94/94             0s 2ms/step -
accuracy: 0.8132 - loss: 0.4091
Epoch 59/100
94/94             0s 2ms/step -
accuracy: 0.8042 - loss: 0.4137
Epoch 60/100
94/94             0s 2ms/step -
accuracy: 0.8223 - loss: 0.3994
Epoch 61/100
94/94             0s 2ms/step -
accuracy: 0.8286 - loss: 0.3947
Epoch 62/100
94/94             0s 2ms/step -
accuracy: 0.8247 - loss: 0.3849
Epoch 63/100
94/94             0s 2ms/step -
accuracy: 0.8243 - loss: 0.3971
Epoch 64/100
94/94             0s 2ms/step -
accuracy: 0.8203 - loss: 0.3942
Epoch 65/100
94/94             0s 2ms/step -
accuracy: 0.8078 - loss: 0.4042
Epoch 66/100
94/94             0s 2ms/step -
accuracy: 0.8351 - loss: 0.3827
Epoch 67/100
94/94             0s 2ms/step -
accuracy: 0.8314 - loss: 0.3818
Epoch 68/100
94/94             0s 2ms/step -
accuracy: 0.8191 - loss: 0.4011
Epoch 69/100
94/94             0s 2ms/step -
accuracy: 0.8286 - loss: 0.3984
Epoch 70/100
94/94             0s 3ms/step -
accuracy: 0.8261 - loss: 0.3881
Epoch 71/100
94/94             0s 2ms/step -
accuracy: 0.8260 - loss: 0.3938
Epoch 72/100
```

```
94/94               0s 2ms/step -
accuracy: 0.8351 - loss: 0.3720
Epoch 73/100
94/94               0s 2ms/step -
accuracy: 0.8254 - loss: 0.3973
Epoch 74/100
94/94               0s 2ms/step -
accuracy: 0.8267 - loss: 0.3848
Epoch 75/100
94/94               0s 2ms/step -
accuracy: 0.8294 - loss: 0.3993
Epoch 76/100
94/94               0s 2ms/step -
accuracy: 0.8218 - loss: 0.3884
Epoch 77/100
94/94               0s 2ms/step -
accuracy: 0.8380 - loss: 0.3812
Epoch 78/100
94/94               0s 2ms/step -
accuracy: 0.8063 - loss: 0.4064
Epoch 79/100
94/94               0s 2ms/step -
accuracy: 0.8320 - loss: 0.3782
Epoch 80/100
94/94               0s 2ms/step -
accuracy: 0.8455 - loss: 0.3629
Epoch 81/100
94/94               0s 2ms/step -
accuracy: 0.8332 - loss: 0.3712
Epoch 82/100
94/94               0s 2ms/step -
accuracy: 0.8205 - loss: 0.3914
Epoch 83/100
94/94               0s 2ms/step -
accuracy: 0.8279 - loss: 0.3762
Epoch 84/100
94/94               0s 2ms/step -
accuracy: 0.8335 - loss: 0.3814
Epoch 85/100
94/94               0s 2ms/step -
accuracy: 0.8368 - loss: 0.3857
Epoch 86/100
94/94               0s 2ms/step -
accuracy: 0.8329 - loss: 0.3675
Epoch 87/100
94/94               0s 2ms/step -
accuracy: 0.8398 - loss: 0.3635
Epoch 88/100
```

```
94/94            0s 2ms/step -
accuracy: 0.8261 - loss: 0.3816
Epoch 89/100
94/94            0s 2ms/step -
accuracy: 0.8288 - loss: 0.3777
Epoch 90/100
94/94            0s 2ms/step -
accuracy: 0.8345 - loss: 0.3805
Epoch 91/100
94/94            0s 2ms/step -
accuracy: 0.8301 - loss: 0.3787
Epoch 92/100
94/94            0s 2ms/step -
accuracy: 0.8422 - loss: 0.3771
Epoch 93/100
94/94            0s 2ms/step -
accuracy: 0.8172 - loss: 0.3875
Epoch 94/100
94/94            0s 2ms/step -
accuracy: 0.8350 - loss: 0.3622
Epoch 95/100
94/94            0s 2ms/step -
accuracy: 0.8239 - loss: 0.3997
Epoch 96/100
94/94            0s 2ms/step -
accuracy: 0.8251 - loss: 0.3868
Epoch 97/100
94/94            0s 2ms/step -
accuracy: 0.8236 - loss: 0.3958
Epoch 98/100
94/94            0s 2ms/step -
accuracy: 0.8434 - loss: 0.3516
Epoch 99/100
94/94            0s 2ms/step -
accuracy: 0.8310 - loss: 0.3640
Epoch 100/100
94/94            0s 2ms/step -
accuracy: 0.8390 - loss: 0.3675
24/24            0s 2ms/step -
accuracy: 0.7638 - loss: 0.5788
[0.5729227066040039, 0.7553476095199585]
24/24            0s 4ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.74      0.75       374
           1       0.75      0.77      0.76       374
```

```
    accuracy                           0.76       748
   macro avg       0.76      0.76      0.76       748
weighted avg       0.76      0.76      0.76       748
```

2.0.1 f1 score is improved in the above cell

3 Method 2: Oversampling

[77]:
```
count_class_0,count_class_1
```

[77]:
```
(5163, 1869)
```

3.1 here only 1869 is available in count class 1 but we need 5163 for this we can simple use .sample with replace=True. This will add duplicate values randomly

[80]:
```
df_class_1_over=df_class_1.sample(count_class_0,replace=True)
df_test_over=pd.concat([df_class_0,df_class_1_over],axis='rows')
```

[81]:
```
df_test_over.shape
```

[81]:
```
(10326, 27)
```

[82]:
```
print("Random over-sampling")
print(df_test_over.Churn.value_counts())
```

```
Random over-sampling
Churn
0    5163
1    5163
Name: count, dtype: int64
```

[83]:
```
x=df_test_over.drop('Churn',axis='columns')
y=df_test_over['Churn']

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
 ↪2,random_state=15,stratify=y)
```

[84]:
```
y_test.value_counts()
```

[84]:
```
Churn
1    1033
0    1033
Name: count, dtype: int64
```

[85]:
```
y_preds=ANN(x_train,y_train,x_test,y_test,'binary_crossentropy',-1)
```

```
Epoch 1/100
```

```
C:\Users\admin\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
259/259              3s 2ms/step -
accuracy: 0.6637 - loss: 0.6029
Epoch 2/100
259/259              1s 2ms/step -
accuracy: 0.7490 - loss: 0.5089
Epoch 3/100
259/259              1s 3ms/step -
accuracy: 0.7699 - loss: 0.4815
Epoch 4/100
259/259              1s 2ms/step -
accuracy: 0.7721 - loss: 0.4729
Epoch 5/100
259/259              1s 2ms/step -
accuracy: 0.7642 - loss: 0.4766
Epoch 6/100
259/259              1s 3ms/step -
accuracy: 0.7707 - loss: 0.4639
Epoch 7/100
259/259              1s 2ms/step -
accuracy: 0.7642 - loss: 0.4781
Epoch 8/100
259/259              1s 2ms/step -
accuracy: 0.7763 - loss: 0.4632
Epoch 9/100
259/259              1s 2ms/step -
accuracy: 0.7863 - loss: 0.4558
Epoch 10/100
259/259              1s 2ms/step -
accuracy: 0.7891 - loss: 0.4518
Epoch 11/100
259/259              1s 2ms/step -
accuracy: 0.7756 - loss: 0.4639
Epoch 12/100
259/259              1s 2ms/step -
accuracy: 0.7813 - loss: 0.4588
Epoch 13/100
259/259              1s 2ms/step -
accuracy: 0.7861 - loss: 0.4496
Epoch 14/100
259/259              1s 2ms/step -
accuracy: 0.7939 - loss: 0.4327
Epoch 15/100
```

```
259/259              1s 2ms/step -
accuracy: 0.7928 - loss: 0.4428
Epoch 16/100
259/259              1s 2ms/step -
accuracy: 0.7903 - loss: 0.4357
Epoch 17/100
259/259              1s 2ms/step -
accuracy: 0.8020 - loss: 0.4308
Epoch 18/100
259/259              1s 3ms/step -
accuracy: 0.7894 - loss: 0.4442
Epoch 19/100
259/259              1s 2ms/step -
accuracy: 0.7963 - loss: 0.4385
Epoch 20/100
259/259              1s 2ms/step -
accuracy: 0.7966 - loss: 0.4339
Epoch 21/100
259/259              1s 2ms/step -
accuracy: 0.8046 - loss: 0.4236
Epoch 22/100
259/259              1s 2ms/step -
accuracy: 0.7966 - loss: 0.4273
Epoch 23/100
259/259              1s 3ms/step -
accuracy: 0.7944 - loss: 0.4374
Epoch 24/100
259/259              1s 2ms/step -
accuracy: 0.8031 - loss: 0.4204
Epoch 25/100
259/259              1s 2ms/step -
accuracy: 0.8004 - loss: 0.4309
Epoch 26/100
259/259              1s 2ms/step -
accuracy: 0.8002 - loss: 0.4325
Epoch 27/100
259/259              1s 2ms/step -
accuracy: 0.8092 - loss: 0.4170
Epoch 28/100
259/259              1s 2ms/step -
accuracy: 0.8090 - loss: 0.4184
Epoch 29/100
259/259              1s 2ms/step -
accuracy: 0.8080 - loss: 0.4175
Epoch 30/100
259/259              1s 3ms/step -
accuracy: 0.8089 - loss: 0.4134
Epoch 31/100
```

```
259/259               1s 2ms/step -
accuracy: 0.8098 - loss: 0.4133
Epoch 32/100
259/259               1s 2ms/step -
accuracy: 0.8126 - loss: 0.4110
Epoch 33/100
259/259               1s 2ms/step -
accuracy: 0.8116 - loss: 0.4114
Epoch 34/100
259/259               1s 2ms/step -
accuracy: 0.8089 - loss: 0.4155
Epoch 35/100
259/259               1s 2ms/step -
accuracy: 0.8051 - loss: 0.4252
Epoch 36/100
259/259               1s 2ms/step -
accuracy: 0.8139 - loss: 0.4082
Epoch 37/100
259/259               1s 3ms/step -
accuracy: 0.8096 - loss: 0.4094
Epoch 38/100
259/259               1s 2ms/step -
accuracy: 0.8072 - loss: 0.4178
Epoch 39/100
259/259               1s 2ms/step -
accuracy: 0.8123 - loss: 0.4071
Epoch 40/100
259/259               1s 2ms/step -
accuracy: 0.8115 - loss: 0.3978
Epoch 41/100
259/259               1s 2ms/step -
accuracy: 0.8156 - loss: 0.3987
Epoch 42/100
259/259               1s 2ms/step -
accuracy: 0.8157 - loss: 0.3990
Epoch 43/100
259/259               1s 3ms/step -
accuracy: 0.8099 - loss: 0.4024
Epoch 44/100
259/259               1s 2ms/step -
accuracy: 0.8220 - loss: 0.3938
Epoch 45/100
259/259               1s 2ms/step -
accuracy: 0.8175 - loss: 0.3918
Epoch 46/100
259/259               1s 2ms/step -
accuracy: 0.8127 - loss: 0.3991
Epoch 47/100
```

```
259/259              1s 3ms/step -
accuracy: 0.8111 - loss: 0.4025
Epoch 48/100
259/259              1s 2ms/step -
accuracy: 0.8215 - loss: 0.3941
Epoch 49/100
259/259              1s 2ms/step -
accuracy: 0.8229 - loss: 0.3903
Epoch 50/100
259/259              1s 2ms/step -
accuracy: 0.8212 - loss: 0.3931
Epoch 51/100
259/259              1s 2ms/step -
accuracy: 0.8151 - loss: 0.4027
Epoch 52/100
259/259              1s 3ms/step -
accuracy: 0.8142 - loss: 0.3944
Epoch 53/100
259/259              1s 2ms/step -
accuracy: 0.8177 - loss: 0.3921
Epoch 54/100
259/259              1s 2ms/step -
accuracy: 0.8209 - loss: 0.3891
Epoch 55/100
259/259              1s 2ms/step -
accuracy: 0.8146 - loss: 0.3880
Epoch 56/100
259/259              1s 3ms/step -
accuracy: 0.8214 - loss: 0.3914
Epoch 57/100
259/259              1s 2ms/step -
accuracy: 0.8260 - loss: 0.3848
Epoch 58/100
259/259              1s 2ms/step -
accuracy: 0.8235 - loss: 0.3855
Epoch 59/100
259/259              1s 3ms/step -
accuracy: 0.8259 - loss: 0.3804
Epoch 60/100
259/259              1s 2ms/step -
accuracy: 0.8287 - loss: 0.3844
Epoch 61/100
259/259              1s 2ms/step -
accuracy: 0.8232 - loss: 0.3854
Epoch 62/100
259/259              1s 2ms/step -
accuracy: 0.8288 - loss: 0.3860
Epoch 63/100
```

```
259/259              1s 2ms/step -
accuracy: 0.8215 - loss: 0.3874
Epoch 64/100
259/259              1s 2ms/step -
accuracy: 0.8243 - loss: 0.3874
Epoch 65/100
259/259              1s 3ms/step -
accuracy: 0.8315 - loss: 0.3747
Epoch 66/100
259/259              1s 2ms/step -
accuracy: 0.8281 - loss: 0.3854
Epoch 67/100
259/259              1s 2ms/step -
accuracy: 0.8268 - loss: 0.3792
Epoch 68/100
259/259              1s 2ms/step -
accuracy: 0.8324 - loss: 0.3754
Epoch 69/100
259/259              1s 2ms/step -
accuracy: 0.8248 - loss: 0.3828
Epoch 70/100
259/259              1s 2ms/step -
accuracy: 0.8266 - loss: 0.3855
Epoch 71/100
259/259              1s 2ms/step -
accuracy: 0.8290 - loss: 0.3818
Epoch 72/100
259/259              1s 2ms/step -
accuracy: 0.8276 - loss: 0.3811
Epoch 73/100
259/259              1s 2ms/step -
accuracy: 0.8342 - loss: 0.3698
Epoch 74/100
259/259              1s 2ms/step -
accuracy: 0.8334 - loss: 0.3691
Epoch 75/100
259/259              1s 2ms/step -
accuracy: 0.8332 - loss: 0.3721
Epoch 76/100
259/259              1s 2ms/step -
accuracy: 0.8289 - loss: 0.3745
Epoch 77/100
259/259              1s 2ms/step -
accuracy: 0.8334 - loss: 0.3714
Epoch 78/100
259/259              1s 2ms/step -
accuracy: 0.8381 - loss: 0.3686
Epoch 79/100
```

```
259/259              1s 2ms/step -
accuracy: 0.8361 - loss: 0.3681
Epoch 80/100
259/259              1s 2ms/step -
accuracy: 0.8389 - loss: 0.3622
Epoch 81/100
259/259              1s 2ms/step -
accuracy: 0.8314 - loss: 0.3701
Epoch 82/100
259/259              1s 2ms/step -
accuracy: 0.8371 - loss: 0.3651
Epoch 83/100
259/259              1s 3ms/step -
accuracy: 0.8316 - loss: 0.3681
Epoch 84/100
259/259              1s 2ms/step -
accuracy: 0.8366 - loss: 0.3619
Epoch 85/100
259/259              1s 2ms/step -
accuracy: 0.8407 - loss: 0.3589
Epoch 86/100
259/259              1s 2ms/step -
accuracy: 0.8338 - loss: 0.3696
Epoch 87/100
259/259              1s 3ms/step -
accuracy: 0.8399 - loss: 0.3594
Epoch 88/100
259/259              1s 2ms/step -
accuracy: 0.8428 - loss: 0.3545
Epoch 89/100
259/259              1s 2ms/step -
accuracy: 0.8372 - loss: 0.3619
Epoch 90/100
259/259              1s 2ms/step -
accuracy: 0.8373 - loss: 0.3616
Epoch 91/100
259/259              1s 2ms/step -
accuracy: 0.8442 - loss: 0.3614
Epoch 92/100
259/259              1s 2ms/step -
accuracy: 0.8346 - loss: 0.3640
Epoch 93/100
259/259              1s 2ms/step -
accuracy: 0.8425 - loss: 0.3602
Epoch 94/100
259/259              1s 2ms/step -
accuracy: 0.8384 - loss: 0.3610
Epoch 95/100
```

```
259/259              1s 2ms/step -
accuracy: 0.8331 - loss: 0.3727
Epoch 96/100
259/259              1s 2ms/step -
accuracy: 0.8375 - loss: 0.3599
Epoch 97/100
259/259              1s 2ms/step -
accuracy: 0.8464 - loss: 0.3528
Epoch 98/100
259/259              1s 2ms/step -
accuracy: 0.8349 - loss: 0.3650
Epoch 99/100
259/259              1s 2ms/step -
accuracy: 0.8457 - loss: 0.3584
Epoch 100/100
259/259              1s 2ms/step -
accuracy: 0.8410 - loss: 0.3595
65/65            0s 2ms/step -
accuracy: 0.7824 - loss: 0.4671
[0.4756731688976288, 0.7783156037330627]
65/65            0s 3ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.73      0.77      1033
           1       0.75      0.83      0.79      1033

    accuracy                           0.78      2066
   macro avg       0.78      0.78      0.78      2066
weighted avg       0.78      0.78      0.78      2066
```

3.1.1 f1 score is incresed

4 Method 3: SMOTE

4.1 Here unlike over sampling we won't duplicate the available samples again and again. Here it will create new samples using k nearest neighbour algorithm for this

[86]:
```
x=df2.drop('Churn',axis='columns')
y=df2['Churn']
```

[88]:
```
y.value_counts()
```

[88]:
```
Churn
0    5163
1    1869
```

```
Name: count, dtype: int64
```

[92]:
```
from imblearn.over_sampling import SMOTE
smote=SMOTE(sampling_strategy='minority')
x_sm,y_sm=smote.fit_resample(x,y)
```

[93]:
```
y_sm.value_counts()
```

[93]:
```
Churn
0    5163
1    5163
Name: count, dtype: int64
```

[94]:
```
x_train,x_test,y_train,y_test=train_test_split(x_sm,y_sm,test_size=0.
↪2,random_state=15,stratify=y_sm)
```

[95]:
```
y_train.value_counts()
```

[95]:
```
Churn
1    4130
0    4130
Name: count, dtype: int64
```

[96]:
```
y_test.value_counts()
```

[96]:
```
Churn
1    1033
0    1033
Name: count, dtype: int64
```

[97]:
```
y_preds=ANN(x_train,y_train,x_test,y_test,'binary_crossentropy',-1)
```

```
Epoch 1/100
```

```
C:\Users\admin\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
259/259             3s 2ms/step -
accuracy: 0.7116 - loss: 0.5798
Epoch 2/100
259/259             1s 2ms/step -
accuracy: 0.7726 - loss: 0.4770
Epoch 3/100
259/259             1s 2ms/step -
accuracy: 0.7776 - loss: 0.4733
Epoch 4/100
```

```
259/259             1s 2ms/step -
accuracy: 0.7946 - loss: 0.4443
Epoch 5/100
259/259             1s 2ms/step -
accuracy: 0.7886 - loss: 0.4474
Epoch 6/100
259/259             1s 2ms/step -
accuracy: 0.7854 - loss: 0.4494
Epoch 7/100
259/259             1s 2ms/step -
accuracy: 0.7883 - loss: 0.4462
Epoch 8/100
259/259             1s 2ms/step -
accuracy: 0.7970 - loss: 0.4360
Epoch 9/100
259/259             1s 2ms/step -
accuracy: 0.7971 - loss: 0.4351
Epoch 10/100
259/259             1s 3ms/step -
accuracy: 0.8115 - loss: 0.4152
Epoch 11/100
259/259             1s 2ms/step -
accuracy: 0.8007 - loss: 0.4235
Epoch 12/100
259/259             1s 2ms/step -
accuracy: 0.8054 - loss: 0.4202
Epoch 13/100
259/259             1s 2ms/step -
accuracy: 0.7971 - loss: 0.4250
Epoch 14/100
259/259             1s 2ms/step -
accuracy: 0.8023 - loss: 0.4155
Epoch 15/100
259/259             1s 2ms/step -
accuracy: 0.8004 - loss: 0.4187
Epoch 16/100
259/259             1s 2ms/step -
accuracy: 0.8097 - loss: 0.4092
Epoch 17/100
259/259             1s 2ms/step -
accuracy: 0.8175 - loss: 0.4037
Epoch 18/100
259/259             1s 2ms/step -
accuracy: 0.8107 - loss: 0.4035
Epoch 19/100
259/259             1s 2ms/step -
accuracy: 0.8114 - loss: 0.4083
Epoch 20/100
```

```
259/259              1s 2ms/step -
accuracy: 0.8151 - loss: 0.4038
Epoch 21/100
259/259              1s 2ms/step -
accuracy: 0.8097 - loss: 0.4104
Epoch 22/100
259/259              1s 2ms/step -
accuracy: 0.8169 - loss: 0.4003
Epoch 23/100
259/259              1s 2ms/step -
accuracy: 0.8231 - loss: 0.3916
Epoch 24/100
259/259              1s 2ms/step -
accuracy: 0.8214 - loss: 0.3944
Epoch 25/100
259/259              1s 2ms/step -
accuracy: 0.8238 - loss: 0.3896
Epoch 26/100
259/259              1s 2ms/step -
accuracy: 0.8231 - loss: 0.3906
Epoch 27/100
259/259              1s 2ms/step -
accuracy: 0.8215 - loss: 0.3964
Epoch 28/100
259/259              1s 2ms/step -
accuracy: 0.8214 - loss: 0.3931
Epoch 29/100
259/259              1s 2ms/step -
accuracy: 0.8225 - loss: 0.3830
Epoch 30/100
259/259              1s 2ms/step -
accuracy: 0.8315 - loss: 0.3744
Epoch 31/100
259/259              1s 2ms/step -
accuracy: 0.8227 - loss: 0.3921
Epoch 32/100
259/259              1s 2ms/step -
accuracy: 0.8264 - loss: 0.3920
Epoch 33/100
259/259              1s 2ms/step -
accuracy: 0.8231 - loss: 0.3811
Epoch 34/100
259/259              1s 2ms/step -
accuracy: 0.8284 - loss: 0.3854
Epoch 35/100
259/259              1s 2ms/step -
accuracy: 0.8411 - loss: 0.3724
Epoch 36/100
```

```
259/259              1s 2ms/step -
accuracy: 0.8333 - loss: 0.3716
Epoch 37/100
259/259              1s 2ms/step -
accuracy: 0.8298 - loss: 0.3700
Epoch 38/100
259/259              1s 2ms/step -
accuracy: 0.8252 - loss: 0.3794
Epoch 39/100
259/259              1s 2ms/step -
accuracy: 0.8278 - loss: 0.3778
Epoch 40/100
259/259              1s 2ms/step -
accuracy: 0.8402 - loss: 0.3670
Epoch 41/100
259/259              1s 2ms/step -
accuracy: 0.8364 - loss: 0.3682
Epoch 42/100
259/259              1s 2ms/step -
accuracy: 0.8342 - loss: 0.3749
Epoch 43/100
259/259              1s 2ms/step -
accuracy: 0.8393 - loss: 0.3738
Epoch 44/100
259/259              1s 2ms/step -
accuracy: 0.8366 - loss: 0.3647
Epoch 45/100
259/259              1s 2ms/step -
accuracy: 0.8441 - loss: 0.3587
Epoch 46/100
259/259              1s 2ms/step -
accuracy: 0.8369 - loss: 0.3674
Epoch 47/100
259/259              1s 2ms/step -
accuracy: 0.8400 - loss: 0.3702
Epoch 48/100
259/259              1s 2ms/step -
accuracy: 0.8393 - loss: 0.3602
Epoch 49/100
259/259              1s 2ms/step -
accuracy: 0.8422 - loss: 0.3601
Epoch 50/100
259/259              1s 2ms/step -
accuracy: 0.8400 - loss: 0.3652
Epoch 51/100
259/259              1s 2ms/step -
accuracy: 0.8418 - loss: 0.3589
Epoch 52/100
```

```
259/259              1s 2ms/step -
accuracy: 0.8365 - loss: 0.3720
Epoch 53/100
259/259              1s 2ms/step -
accuracy: 0.8449 - loss: 0.3550
Epoch 54/100
259/259              1s 2ms/step -
accuracy: 0.8400 - loss: 0.3667
Epoch 55/100
259/259              1s 2ms/step -
accuracy: 0.8422 - loss: 0.3619
Epoch 56/100
259/259              1s 2ms/step -
accuracy: 0.8386 - loss: 0.3653
Epoch 57/100
259/259              1s 2ms/step -
accuracy: 0.8470 - loss: 0.3569
Epoch 58/100
259/259              1s 2ms/step -
accuracy: 0.8500 - loss: 0.3476
Epoch 59/100
259/259              1s 2ms/step -
accuracy: 0.8408 - loss: 0.3614
Epoch 60/100
259/259              1s 2ms/step -
accuracy: 0.8406 - loss: 0.3575
Epoch 61/100
259/259              1s 2ms/step -
accuracy: 0.8466 - loss: 0.3470
Epoch 62/100
259/259              1s 2ms/step -
accuracy: 0.8445 - loss: 0.3562
Epoch 63/100
259/259              1s 2ms/step -
accuracy: 0.8384 - loss: 0.3634
Epoch 64/100
259/259              1s 2ms/step -
accuracy: 0.8433 - loss: 0.3582
Epoch 65/100
259/259              1s 2ms/step -
accuracy: 0.8500 - loss: 0.3457
Epoch 66/100
259/259              1s 2ms/step -
accuracy: 0.8499 - loss: 0.3499
Epoch 67/100
259/259              1s 2ms/step -
accuracy: 0.8544 - loss: 0.3477
Epoch 68/100
```

```
259/259              1s 2ms/step -
accuracy: 0.8445 - loss: 0.3512
Epoch 69/100
259/259              1s 2ms/step -
accuracy: 0.8534 - loss: 0.3467
Epoch 70/100
259/259              1s 2ms/step -
accuracy: 0.8456 - loss: 0.3539
Epoch 71/100
259/259              1s 2ms/step -
accuracy: 0.8425 - loss: 0.3565
Epoch 72/100
259/259              1s 2ms/step -
accuracy: 0.8419 - loss: 0.3509
Epoch 73/100
259/259              1s 2ms/step -
accuracy: 0.8468 - loss: 0.3485
Epoch 74/100
259/259              1s 2ms/step -
accuracy: 0.8534 - loss: 0.3459
Epoch 75/100
259/259              1s 2ms/step -
accuracy: 0.8523 - loss: 0.3430
Epoch 76/100
259/259              1s 2ms/step -
accuracy: 0.8472 - loss: 0.3524
Epoch 77/100
259/259              1s 2ms/step -
accuracy: 0.8529 - loss: 0.3496
Epoch 78/100
259/259              1s 2ms/step -
accuracy: 0.8509 - loss: 0.3436
Epoch 79/100
259/259              1s 2ms/step -
accuracy: 0.8497 - loss: 0.3457
Epoch 80/100
259/259              1s 2ms/step -
accuracy: 0.8486 - loss: 0.3443
Epoch 81/100
259/259              1s 2ms/step -
accuracy: 0.8528 - loss: 0.3448
Epoch 82/100
259/259              1s 2ms/step -
accuracy: 0.8518 - loss: 0.3443
Epoch 83/100
259/259              1s 2ms/step -
accuracy: 0.8452 - loss: 0.3451
Epoch 84/100
```

```
259/259              1s 2ms/step -
accuracy: 0.8473 - loss: 0.3463
Epoch 85/100
259/259              1s 2ms/step -
accuracy: 0.8492 - loss: 0.3507
Epoch 86/100
259/259              1s 2ms/step -
accuracy: 0.8509 - loss: 0.3518
Epoch 87/100
259/259              1s 2ms/step -
accuracy: 0.8489 - loss: 0.3497
Epoch 88/100
259/259              1s 2ms/step -
accuracy: 0.8506 - loss: 0.3420
Epoch 89/100
259/259              1s 2ms/step -
accuracy: 0.8568 - loss: 0.3416
Epoch 90/100
259/259              1s 2ms/step -
accuracy: 0.8575 - loss: 0.3338
Epoch 91/100
259/259              1s 2ms/step -
accuracy: 0.8488 - loss: 0.3465
Epoch 92/100
259/259              1s 2ms/step -
accuracy: 0.8555 - loss: 0.3389
Epoch 93/100
259/259              1s 2ms/step -
accuracy: 0.8541 - loss: 0.3395
Epoch 94/100
259/259              1s 2ms/step -
accuracy: 0.8525 - loss: 0.3365
Epoch 95/100
259/259              1s 2ms/step -
accuracy: 0.8593 - loss: 0.3323
Epoch 96/100
259/259              1s 2ms/step -
accuracy: 0.8572 - loss: 0.3336
Epoch 97/100
259/259              1s 2ms/step -
accuracy: 0.8543 - loss: 0.3394
Epoch 98/100
259/259              1s 2ms/step -
accuracy: 0.8465 - loss: 0.3492
Epoch 99/100
259/259              1s 2ms/step -
accuracy: 0.8541 - loss: 0.3371
Epoch 100/100
```

```
259/259            1s 2ms/step -
accuracy: 0.8566 - loss: 0.3331
65/65             0s 2ms/step -
accuracy: 0.7993 - loss: 0.4367
[0.4392593801021576, 0.7981606721878052]
65/65             0s 3ms/step
Classification Report:
               precision    recall  f1-score   support

           0       0.78      0.84      0.81      1033
           1       0.82      0.76      0.79      1033

    accuracy                           0.80      2066
   macro avg       0.80      0.80      0.80      2066
weighted avg       0.80      0.80      0.80      2066
```

5 Method 4: Use of Ensemble with undersampling

[98]:
```
df2.Churn.value_counts()
```

[98]:
```
Churn
0    5163
1    1869
Name: count, dtype: int64
```

[99]:
```
x=df2.drop('Churn',axis='columns')
y=df2['Churn']
```

[100]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.
 ↪2,random_state=15,stratify=y)
```

[101]:
```
y_train.value_counts()
```

[101]:
```
Churn
0    4130
1    1495
Name: count, dtype: int64
```

[103]:
```
4130/1495 #we will divide it into three batches
```

[103]: 2.762541806020067

[104]:
```
4130/3
```

[104]: 1376.6666666666667

[107]:
```
df3=x_train.copy()
df3['Churn']=y_train
```

[110]:
```
df3_class0=df3[df3['Churn']==0]
df3_class1=df3[df3['Churn']==1]
```

[111]:
```
df3_class0.shape,df3_class1.shape
```

[111]:
```
((4130, 27), (1495, 27))
```

[112]:
```
df3_class0[:1495].shape
```

[112]:
```
(1495, 27)
```

[120]:
```
def get_train_batch(df_majority,df_minority,start,end):
    df_train=pd.concat([df_majority[start:end],df_minority],axis=0)
    x_train=df_train.drop('Churn',axis='columns')
    y_train=df_train.Churn
    return x_train,y_train
```

[128]:
```
x_train,y_train=get_train_batch(df3_class0,df3_class1,0,1495)
y_pred1=ANN(x_train,y_train,x_test,y_test,'binary_crossentropy',-1)
```

```
Epoch 1/100

C:\Users\admin\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)

94/94                 1s 960us/step -
accuracy: 0.5884 - loss: 0.6700
Epoch 2/100
94/94                 0s 876us/step -
accuracy: 0.7522 - loss: 0.5319
Epoch 3/100
94/94                 0s 902us/step -
accuracy: 0.7873 - loss: 0.4735
Epoch 4/100
94/94                 0s 866us/step -
accuracy: 0.7738 - loss: 0.4838
Epoch 5/100
94/94                 0s 917us/step -
accuracy: 0.7422 - loss: 0.5137
Epoch 6/100
94/94                 0s 949us/step -
accuracy: 0.7761 - loss: 0.4659
Epoch 7/100
```

```
94/94            0s 899us/step -
accuracy: 0.7860 - loss: 0.4600
Epoch 8/100
94/94            0s 897us/step -
accuracy: 0.7812 - loss: 0.4768
Epoch 9/100
94/94            0s 1ms/step -
accuracy: 0.7903 - loss: 0.4709
Epoch 10/100
94/94            0s 941us/step -
accuracy: 0.7936 - loss: 0.4543
Epoch 11/100
94/94            0s 941us/step -
accuracy: 0.7770 - loss: 0.4792
Epoch 12/100
94/94            0s 977us/step -
accuracy: 0.7767 - loss: 0.4783
Epoch 13/100
94/94            0s 1ms/step -
accuracy: 0.7857 - loss: 0.4634
Epoch 14/100
94/94            0s 903us/step -
accuracy: 0.7809 - loss: 0.4791
Epoch 15/100
94/94            0s 974us/step -
accuracy: 0.7989 - loss: 0.4469
Epoch 16/100
94/94            0s 2ms/step -
accuracy: 0.7836 - loss: 0.4650
Epoch 17/100
94/94            0s 902us/step -
accuracy: 0.7780 - loss: 0.4667
Epoch 18/100
94/94            0s 1ms/step -
accuracy: 0.7949 - loss: 0.4324
Epoch 19/100
94/94            0s 886us/step -
accuracy: 0.7936 - loss: 0.4631
Epoch 20/100
94/94            0s 848us/step -
accuracy: 0.7895 - loss: 0.4557
Epoch 21/100
94/94            0s 855us/step -
accuracy: 0.7887 - loss: 0.4504
Epoch 22/100
94/94            0s 976us/step -
accuracy: 0.7859 - loss: 0.4577
Epoch 23/100
```

```
94/94            0s 940us/step -
accuracy: 0.7865 - loss: 0.4474
Epoch 24/100
94/94            0s 907us/step -
accuracy: 0.7871 - loss: 0.4624
Epoch 25/100
94/94            0s 902us/step -
accuracy: 0.7917 - loss: 0.4442
Epoch 26/100
94/94            0s 941us/step -
accuracy: 0.7805 - loss: 0.4645
Epoch 27/100
94/94            0s 1ms/step -
accuracy: 0.7940 - loss: 0.4486
Epoch 28/100
94/94            0s 946us/step -
accuracy: 0.7756 - loss: 0.4670
Epoch 29/100
94/94            0s 944us/step -
accuracy: 0.7916 - loss: 0.4572
Epoch 30/100
94/94            0s 945us/step -
accuracy: 0.7928 - loss: 0.4473
Epoch 31/100
94/94            0s 906us/step -
accuracy: 0.7926 - loss: 0.4457
Epoch 32/100
94/94            0s 894us/step -
accuracy: 0.7938 - loss: 0.4480
Epoch 33/100
94/94            0s 890us/step -
accuracy: 0.7888 - loss: 0.4494
Epoch 34/100
94/94            0s 1ms/step -
accuracy: 0.7971 - loss: 0.4402
Epoch 35/100
94/94            0s 944us/step -
accuracy: 0.7948 - loss: 0.4409
Epoch 36/100
94/94            0s 890us/step -
accuracy: 0.7951 - loss: 0.4461
Epoch 37/100
94/94            0s 1ms/step -
accuracy: 0.7961 - loss: 0.4400
Epoch 38/100
94/94            0s 983us/step -
accuracy: 0.8038 - loss: 0.4403
Epoch 39/100
```

```
94/94            0s 981us/step -
accuracy: 0.7932 - loss: 0.4484
Epoch 40/100
94/94            0s 964us/step -
accuracy: 0.7826 - loss: 0.4580
Epoch 41/100
94/94            0s 851us/step -
accuracy: 0.7972 - loss: 0.4330
Epoch 42/100
94/94            0s 869us/step -
accuracy: 0.7878 - loss: 0.4553
Epoch 43/100
94/94            0s 919us/step -
accuracy: 0.8049 - loss: 0.4292
Epoch 44/100
94/94            0s 955us/step -
accuracy: 0.8000 - loss: 0.4435
Epoch 45/100
94/94            0s 912us/step -
accuracy: 0.8042 - loss: 0.4307
Epoch 46/100
94/94            0s 912us/step -
accuracy: 0.7991 - loss: 0.4404
Epoch 47/100
94/94            0s 954us/step -
accuracy: 0.7933 - loss: 0.4337
Epoch 48/100
94/94            0s 1ms/step -
accuracy: 0.7865 - loss: 0.4603
Epoch 49/100
94/94            0s 969us/step -
accuracy: 0.7967 - loss: 0.4267
Epoch 50/100
94/94            0s 2ms/step -
accuracy: 0.7954 - loss: 0.4405
Epoch 51/100
94/94            0s 947us/step -
accuracy: 0.7872 - loss: 0.4327
Epoch 52/100
94/94            0s 962us/step -
accuracy: 0.7897 - loss: 0.4346
Epoch 53/100
94/94            0s 894us/step -
accuracy: 0.7792 - loss: 0.4563
Epoch 54/100
94/94            0s 962us/step -
accuracy: 0.7941 - loss: 0.4386
Epoch 55/100
```

```
94/94            0s 931us/step -
accuracy: 0.8038 - loss: 0.4346
Epoch 56/100
94/94            0s 871us/step -
accuracy: 0.7971 - loss: 0.4274
Epoch 57/100
94/94            0s 876us/step -
accuracy: 0.7879 - loss: 0.4275
Epoch 58/100
94/94            0s 879us/step -
accuracy: 0.7890 - loss: 0.4434
Epoch 59/100
94/94            0s 903us/step -
accuracy: 0.7981 - loss: 0.4289
Epoch 60/100
94/94            0s 937us/step -
accuracy: 0.8054 - loss: 0.4315
Epoch 61/100
94/94            0s 1ms/step -
accuracy: 0.8026 - loss: 0.4284
Epoch 62/100
94/94            0s 1ms/step -
accuracy: 0.8023 - loss: 0.4219
Epoch 63/100
94/94            0s 1ms/step -
accuracy: 0.8067 - loss: 0.4110
Epoch 64/100
94/94            0s 1ms/step -
accuracy: 0.7929 - loss: 0.4239
Epoch 65/100
94/94            0s 1ms/step -
accuracy: 0.8012 - loss: 0.4257
Epoch 66/100
94/94            0s 908us/step -
accuracy: 0.7957 - loss: 0.4230
Epoch 67/100
94/94            0s 972us/step -
accuracy: 0.8066 - loss: 0.4268
Epoch 68/100
94/94            0s 911us/step -
accuracy: 0.8069 - loss: 0.4219
Epoch 69/100
94/94            0s 1ms/step -
accuracy: 0.8027 - loss: 0.4144
Epoch 70/100
94/94            0s 934us/step -
accuracy: 0.8211 - loss: 0.4090
Epoch 71/100
```

```
94/94            0s 927us/step -
accuracy: 0.7959 - loss: 0.4225
Epoch 72/100
94/94            0s 955us/step -
accuracy: 0.8070 - loss: 0.4129
Epoch 73/100
94/94            0s 1ms/step -
accuracy: 0.8059 - loss: 0.4097
Epoch 74/100
94/94            0s 993us/step -
accuracy: 0.7968 - loss: 0.4263
Epoch 75/100
94/94            0s 1ms/step -
accuracy: 0.7946 - loss: 0.4316
Epoch 76/100
94/94            0s 987us/step -
accuracy: 0.8031 - loss: 0.4131
Epoch 77/100
94/94            0s 1ms/step -
accuracy: 0.8057 - loss: 0.4134
Epoch 78/100
94/94            0s 993us/step -
accuracy: 0.8000 - loss: 0.4100
Epoch 79/100
94/94            0s 1ms/step -
accuracy: 0.8006 - loss: 0.4239
Epoch 80/100
94/94            0s 982us/step -
accuracy: 0.8184 - loss: 0.3943
Epoch 81/100
94/94            0s 931us/step -
accuracy: 0.8147 - loss: 0.4107
Epoch 82/100
94/94            0s 953us/step -
accuracy: 0.8038 - loss: 0.4164
Epoch 83/100
94/94            0s 930us/step -
accuracy: 0.8085 - loss: 0.4128
Epoch 84/100
94/94            0s 975us/step -
accuracy: 0.7994 - loss: 0.4175
Epoch 85/100
94/94            0s 985us/step -
accuracy: 0.8095 - loss: 0.4066
Epoch 86/100
94/94            0s 910us/step -
accuracy: 0.8112 - loss: 0.4050
Epoch 87/100
```

```
94/94              0s 902us/step -
accuracy: 0.8078 - loss: 0.4148
Epoch 88/100
94/94              0s 936us/step -
accuracy: 0.8044 - loss: 0.4175
Epoch 89/100
94/94              0s 1ms/step -
accuracy: 0.8081 - loss: 0.4071
Epoch 90/100
94/94              0s 1ms/step -
accuracy: 0.8127 - loss: 0.4085
Epoch 91/100
94/94              0s 1ms/step -
accuracy: 0.8116 - loss: 0.4056
Epoch 92/100
94/94              0s 1ms/step -
accuracy: 0.8241 - loss: 0.3880
Epoch 93/100
94/94              0s 1ms/step -
accuracy: 0.8038 - loss: 0.4135
Epoch 94/100
94/94              0s 2ms/step -
accuracy: 0.8072 - loss: 0.4089
Epoch 95/100
94/94              0s 961us/step -
accuracy: 0.8138 - loss: 0.3994
Epoch 96/100
94/94              0s 1ms/step -
accuracy: 0.8139 - loss: 0.3978
Epoch 97/100
94/94              0s 980us/step -
accuracy: 0.8177 - loss: 0.4003
Epoch 98/100
94/94              0s 1ms/step -
accuracy: 0.8096 - loss: 0.3993
Epoch 99/100
94/94              0s 1ms/step -
accuracy: 0.8252 - loss: 0.3912
Epoch 100/100
94/94              0s 933us/step -
accuracy: 0.8038 - loss: 0.4114
44/44              0s 850us/step -
accuracy: 0.7566 - loss: 0.5190
[0.5366734266281128, 0.7420042753219604]
44/44              0s 2ms/step
Classification Report:
               precision    recall  f1-score   support
```

```
           0       0.89      0.74      0.81      1033
           1       0.51      0.75      0.61       374

    accuracy                           0.74      1407
   macro avg       0.70      0.74      0.71      1407
weighted avg       0.79      0.74      0.75      1407
```

[129]:
```python
x_train,y_train=get_train_batch(df3_class0,df3_class1,1495,2990)
y_pred2=ANN(x_train,y_train,x_test,y_test,'binary_crossentropy',-1)
```

```
Epoch 1/100
```

```
C:\Users\admin\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
94/94                 1s 941us/step -
accuracy: 0.6316 - loss: 0.6421
Epoch 2/100
94/94                 0s 871us/step -
accuracy: 0.7470 - loss: 0.5223
Epoch 3/100
94/94                 0s 879us/step -
accuracy: 0.7534 - loss: 0.5066
Epoch 4/100
94/94                 0s 863us/step -
accuracy: 0.7471 - loss: 0.5004
Epoch 5/100
94/94                 0s 835us/step -
accuracy: 0.7566 - loss: 0.4882
Epoch 6/100
94/94                 0s 831us/step -
accuracy: 0.7603 - loss: 0.4854
Epoch 7/100
94/94                 0s 855us/step -
accuracy: 0.7677 - loss: 0.4777
Epoch 8/100
94/94                 0s 801us/step -
accuracy: 0.7703 - loss: 0.4720
Epoch 9/100
94/94                 0s 841us/step -
accuracy: 0.7693 - loss: 0.4668
Epoch 10/100
94/94                 0s 892us/step -
accuracy: 0.7663 - loss: 0.4664
Epoch 11/100
```

```
94/94            0s 833us/step -
accuracy: 0.7643 - loss: 0.4680
Epoch 12/100
94/94            0s 865us/step -
accuracy: 0.7677 - loss: 0.4811
Epoch 13/100
94/94            0s 888us/step -
accuracy: 0.7759 - loss: 0.4647
Epoch 14/100
94/94            0s 867us/step -
accuracy: 0.7677 - loss: 0.4665
Epoch 15/100
94/94            0s 884us/step -
accuracy: 0.7658 - loss: 0.4690
Epoch 16/100
94/94            0s 820us/step -
accuracy: 0.7737 - loss: 0.4588
Epoch 17/100
94/94            0s 988us/step -
accuracy: 0.7730 - loss: 0.4562
Epoch 18/100
94/94            0s 816us/step -
accuracy: 0.7667 - loss: 0.4689
Epoch 19/100
94/94            0s 889us/step -
accuracy: 0.7784 - loss: 0.4664
Epoch 20/100
94/94            0s 921us/step -
accuracy: 0.7786 - loss: 0.4600
Epoch 21/100
94/94            0s 934us/step -
accuracy: 0.7695 - loss: 0.4558
Epoch 22/100
94/94            0s 881us/step -
accuracy: 0.7725 - loss: 0.4655
Epoch 23/100
94/94            0s 854us/step -
accuracy: 0.7758 - loss: 0.4583
Epoch 24/100
94/94            0s 939us/step -
accuracy: 0.7813 - loss: 0.4483
Epoch 25/100
94/94            0s 848us/step -
accuracy: 0.7721 - loss: 0.4637
Epoch 26/100
94/94            0s 907us/step -
accuracy: 0.7753 - loss: 0.4619
Epoch 27/100
```

```
94/94              0s 856us/step -
accuracy: 0.7743 - loss: 0.4523
Epoch 28/100
94/94              0s 861us/step -
accuracy: 0.7808 - loss: 0.4506
Epoch 29/100
94/94              0s 875us/step -
accuracy: 0.7882 - loss: 0.4417
Epoch 30/100
94/94              0s 878us/step -
accuracy: 0.7765 - loss: 0.4539
Epoch 31/100
94/94              0s 927us/step -
accuracy: 0.7820 - loss: 0.4386
Epoch 32/100
94/94              0s 928us/step -
accuracy: 0.7942 - loss: 0.4318
Epoch 33/100
94/94              0s 874us/step -
accuracy: 0.7955 - loss: 0.4356
Epoch 34/100
94/94              0s 865us/step -
accuracy: 0.7776 - loss: 0.4386
Epoch 35/100
94/94              0s 881us/step -
accuracy: 0.7920 - loss: 0.4371
Epoch 36/100
94/94              0s 857us/step -
accuracy: 0.7831 - loss: 0.4397
Epoch 37/100
94/94              0s 877us/step -
accuracy: 0.7877 - loss: 0.4429
Epoch 38/100
94/94              0s 901us/step -
accuracy: 0.7798 - loss: 0.4515
Epoch 39/100
94/94              0s 997us/step -
accuracy: 0.7954 - loss: 0.4164
Epoch 40/100
94/94              0s 1ms/step -
accuracy: 0.7831 - loss: 0.4425
Epoch 41/100
94/94              0s 875us/step -
accuracy: 0.7933 - loss: 0.4262
Epoch 42/100
94/94              0s 894us/step -
accuracy: 0.7911 - loss: 0.4385
Epoch 43/100
```

```
94/94               0s 898us/step -
accuracy: 0.8030 - loss: 0.4287
Epoch 44/100
94/94               0s 896us/step -
accuracy: 0.7983 - loss: 0.4259
Epoch 45/100
94/94               0s 859us/step -
accuracy: 0.7912 - loss: 0.4322
Epoch 46/100
94/94               0s 889us/step -
accuracy: 0.7957 - loss: 0.4267
Epoch 47/100
94/94               0s 900us/step -
accuracy: 0.7945 - loss: 0.4240
Epoch 48/100
94/94               0s 843us/step -
accuracy: 0.7895 - loss: 0.4405
Epoch 49/100
94/94               0s 870us/step -
accuracy: 0.7989 - loss: 0.4303
Epoch 50/100
94/94               0s 908us/step -
accuracy: 0.7911 - loss: 0.4297
Epoch 51/100
94/94               0s 891us/step -
accuracy: 0.8035 - loss: 0.4118
Epoch 52/100
94/94               0s 901us/step -
accuracy: 0.8004 - loss: 0.4311
Epoch 53/100
94/94               0s 923us/step -
accuracy: 0.7980 - loss: 0.4252
Epoch 54/100
94/94               0s 857us/step -
accuracy: 0.7963 - loss: 0.4230
Epoch 55/100
94/94               0s 865us/step -
accuracy: 0.8082 - loss: 0.4108
Epoch 56/100
94/94               0s 839us/step -
accuracy: 0.8063 - loss: 0.4182
Epoch 57/100
94/94               0s 868us/step -
accuracy: 0.8090 - loss: 0.4089
Epoch 58/100
94/94               0s 970us/step -
accuracy: 0.8101 - loss: 0.4183
Epoch 59/100
```

```
94/94            0s 1ms/step -
accuracy: 0.8039 - loss: 0.4204
Epoch 60/100
94/94            0s 829us/step -
accuracy: 0.7911 - loss: 0.4270
Epoch 61/100
94/94            0s 853us/step -
accuracy: 0.8036 - loss: 0.4189
Epoch 62/100
94/94            0s 843us/step -
accuracy: 0.8054 - loss: 0.4169
Epoch 63/100
94/94            0s 938us/step -
accuracy: 0.8049 - loss: 0.4117
Epoch 64/100
94/94            0s 943us/step -
accuracy: 0.8130 - loss: 0.4049
Epoch 65/100
94/94            0s 865us/step -
accuracy: 0.8166 - loss: 0.3920
Epoch 66/100
94/94            0s 949us/step -
accuracy: 0.8053 - loss: 0.4072
Epoch 67/100
94/94            0s 926us/step -
accuracy: 0.8033 - loss: 0.4215
Epoch 68/100
94/94            0s 871us/step -
accuracy: 0.7974 - loss: 0.4132
Epoch 69/100
94/94            0s 879us/step -
accuracy: 0.7930 - loss: 0.4127
Epoch 70/100
94/94            0s 882us/step -
accuracy: 0.8194 - loss: 0.3967
Epoch 71/100
94/94            0s 876us/step -
accuracy: 0.8108 - loss: 0.4037
Epoch 72/100
94/94            0s 908us/step -
accuracy: 0.8091 - loss: 0.4133
Epoch 73/100
94/94            0s 2ms/step -
accuracy: 0.7959 - loss: 0.4196
Epoch 74/100
94/94            0s 866us/step -
accuracy: 0.8150 - loss: 0.3991
Epoch 75/100
```

```
94/94            0s 864us/step -
accuracy: 0.8148 - loss: 0.3915
Epoch 76/100
94/94            0s 872us/step -
accuracy: 0.8155 - loss: 0.3957
Epoch 77/100
94/94            0s 874us/step -
accuracy: 0.8127 - loss: 0.4061
Epoch 78/100
94/94            0s 872us/step -
accuracy: 0.8091 - loss: 0.4039
Epoch 79/100
94/94            0s 976us/step -
accuracy: 0.8080 - loss: 0.4081
Epoch 80/100
94/94            0s 871us/step -
accuracy: 0.8101 - loss: 0.4038
Epoch 81/100
94/94            0s 865us/step -
accuracy: 0.8158 - loss: 0.3959
Epoch 82/100
94/94            0s 961us/step -
accuracy: 0.8094 - loss: 0.4027
Epoch 83/100
94/94            0s 893us/step -
accuracy: 0.8286 - loss: 0.3905
Epoch 84/100
94/94            0s 922us/step -
accuracy: 0.8135 - loss: 0.4101
Epoch 85/100
94/94            0s 884us/step -
accuracy: 0.8231 - loss: 0.3816
Epoch 86/100
94/94            0s 2ms/step -
accuracy: 0.8320 - loss: 0.3926
Epoch 87/100
94/94            0s 972us/step -
accuracy: 0.8195 - loss: 0.3910
Epoch 88/100
94/94            0s 876us/step -
accuracy: 0.8220 - loss: 0.3907
Epoch 89/100
94/94            0s 855us/step -
accuracy: 0.8164 - loss: 0.3909
Epoch 90/100
94/94            0s 897us/step -
accuracy: 0.8171 - loss: 0.3836
Epoch 91/100
```

```
94/94              0s 889us/step -
accuracy: 0.8137 - loss: 0.3989
Epoch 92/100
94/94              0s 898us/step -
accuracy: 0.8182 - loss: 0.3938
Epoch 93/100
94/94              0s 860us/step -
accuracy: 0.8153 - loss: 0.3974
Epoch 94/100
94/94              0s 881us/step -
accuracy: 0.8152 - loss: 0.3908
Epoch 95/100
94/94              0s 869us/step -
accuracy: 0.8267 - loss: 0.3827
Epoch 96/100
94/94              0s 874us/step -
accuracy: 0.8161 - loss: 0.3867
Epoch 97/100
94/94              0s 2ms/step -
accuracy: 0.8213 - loss: 0.3954
Epoch 98/100
94/94              0s 897us/step -
accuracy: 0.8190 - loss: 0.3954
Epoch 99/100
94/94              0s 1ms/step -
accuracy: 0.8212 - loss: 0.3821
Epoch 100/100
94/94              0s 939us/step -
accuracy: 0.8246 - loss: 0.3780
44/44              0s 777us/step -
accuracy: 0.7367 - loss: 0.5426
[0.5782884359359741, 0.708599865436554]
44/44              0s 2ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.68      0.77      1033
           1       0.47      0.78      0.59       374

    accuracy                           0.71      1407
   macro avg       0.68      0.73      0.68      1407
weighted avg       0.78      0.71      0.72      1407
```

[130]:

```
x_train,y_train=get_train_batch(df3_class0,df3_class1,2990,4130)
y_pred3=ANN(x_train,y_train,x_test,y_test,'binary_crossentropy',-1)
```

```
Epoch 1/100
```

```
C:\Users\admin\AppData\Local\Programs\Python\Python312\Lib\site-
packages\keras\src\layers\core\dense.py:87: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
83/83            1s 826us/step -
accuracy: 0.7007 - loss: 0.6149
Epoch 2/100
83/83            0s 853us/step -
accuracy: 0.7669 - loss: 0.4905
Epoch 3/100
83/83            0s 818us/step -
accuracy: 0.7748 - loss: 0.4864
Epoch 4/100
83/83            0s 794us/step -
accuracy: 0.7663 - loss: 0.4893
Epoch 5/100
83/83            0s 840us/step -
accuracy: 0.7667 - loss: 0.4866
Epoch 6/100
83/83            0s 799us/step -
accuracy: 0.7942 - loss: 0.4702
Epoch 7/100
83/83            0s 814us/step -
accuracy: 0.7779 - loss: 0.4824
Epoch 8/100
83/83            0s 813us/step -
accuracy: 0.7849 - loss: 0.4733
Epoch 9/100
83/83            0s 856us/step -
accuracy: 0.7784 - loss: 0.4728
Epoch 10/100
83/83            0s 852us/step -
accuracy: 0.7833 - loss: 0.4529
Epoch 11/100
83/83            0s 888us/step -
accuracy: 0.7861 - loss: 0.4651
Epoch 12/100
83/83            0s 908us/step -
accuracy: 0.7934 - loss: 0.4481
Epoch 13/100
83/83            0s 901us/step -
accuracy: 0.7941 - loss: 0.4557
Epoch 14/100
83/83            0s 869us/step -
accuracy: 0.7915 - loss: 0.4545
Epoch 15/100
```

```
83/83             0s 870us/step -
accuracy: 0.7969 - loss: 0.4445
Epoch 16/100
83/83             0s 850us/step -
accuracy: 0.7942 - loss: 0.4448
Epoch 17/100
83/83             0s 836us/step -
accuracy: 0.7920 - loss: 0.4549
Epoch 18/100
83/83             0s 845us/step -
accuracy: 0.8018 - loss: 0.4371
Epoch 19/100
83/83             0s 905us/step -
accuracy: 0.7805 - loss: 0.4579
Epoch 20/100
83/83             0s 1ms/step -
accuracy: 0.7911 - loss: 0.4534
Epoch 21/100
83/83             0s 896us/step -
accuracy: 0.7958 - loss: 0.4421
Epoch 22/100
83/83             0s 861us/step -
accuracy: 0.7891 - loss: 0.4449
Epoch 23/100
83/83             0s 856us/step -
accuracy: 0.7982 - loss: 0.4342
Epoch 24/100
83/83             0s 853us/step -
accuracy: 0.7857 - loss: 0.4476
Epoch 25/100
83/83             0s 850us/step -
accuracy: 0.7957 - loss: 0.4435
Epoch 26/100
83/83             0s 925us/step -
accuracy: 0.7908 - loss: 0.4427
Epoch 27/100
83/83             0s 883us/step -
accuracy: 0.8038 - loss: 0.4369
Epoch 28/100
83/83             0s 863us/step -
accuracy: 0.7896 - loss: 0.4462
Epoch 29/100
83/83             0s 847us/step -
accuracy: 0.7973 - loss: 0.4356
Epoch 30/100
83/83             0s 940us/step -
accuracy: 0.7923 - loss: 0.4404
Epoch 31/100
```

```
83/83            0s 894us/step -
accuracy: 0.7905 - loss: 0.4327
Epoch 32/100
83/83            0s 2ms/step -
accuracy: 0.7911 - loss: 0.4422
Epoch 33/100
83/83            0s 899us/step -
accuracy: 0.8039 - loss: 0.4329
Epoch 34/100
83/83            0s 861us/step -
accuracy: 0.7985 - loss: 0.4258
Epoch 35/100
83/83            0s 911us/step -
accuracy: 0.8086 - loss: 0.4172
Epoch 36/100
83/83            0s 940us/step -
accuracy: 0.7978 - loss: 0.4283
Epoch 37/100
83/83            0s 1ms/step -
accuracy: 0.8142 - loss: 0.4113
Epoch 38/100
83/83            0s 855us/step -
accuracy: 0.7947 - loss: 0.4255
Epoch 39/100
83/83            0s 864us/step -
accuracy: 0.8070 - loss: 0.4122
Epoch 40/100
83/83            0s 972us/step -
accuracy: 0.8019 - loss: 0.4264
Epoch 41/100
83/83            0s 845us/step -
accuracy: 0.7958 - loss: 0.4411
Epoch 42/100
83/83            0s 961us/step -
accuracy: 0.7984 - loss: 0.4257
Epoch 43/100
83/83            0s 955us/step -
accuracy: 0.7914 - loss: 0.4297
Epoch 44/100
83/83            0s 898us/step -
accuracy: 0.8063 - loss: 0.4220
Epoch 45/100
83/83            0s 925us/step -
accuracy: 0.8095 - loss: 0.4125
Epoch 46/100
83/83            0s 2ms/step -
accuracy: 0.8068 - loss: 0.4225
Epoch 47/100
```

```
83/83          0s 951us/step -
accuracy: 0.8153 - loss: 0.4127
Epoch 48/100
83/83          0s 917us/step -
accuracy: 0.8130 - loss: 0.4076
Epoch 49/100
83/83          0s 895us/step -
accuracy: 0.8056 - loss: 0.4111
Epoch 50/100
83/83          0s 862us/step -
accuracy: 0.8151 - loss: 0.4029
Epoch 51/100
83/83          0s 1ms/step -
accuracy: 0.8121 - loss: 0.4237
Epoch 52/100
83/83          0s 838us/step -
accuracy: 0.8229 - loss: 0.3900
Epoch 53/100
83/83          0s 890us/step -
accuracy: 0.8048 - loss: 0.4128
Epoch 54/100
83/83          0s 888us/step -
accuracy: 0.8147 - loss: 0.3991
Epoch 55/100
83/83          0s 901us/step -
accuracy: 0.8135 - loss: 0.4063
Epoch 56/100
83/83          0s 1000us/step -
accuracy: 0.8019 - loss: 0.4089
Epoch 57/100
83/83          0s 865us/step -
accuracy: 0.7984 - loss: 0.4232
Epoch 58/100
83/83          0s 871us/step -
accuracy: 0.8151 - loss: 0.4047
Epoch 59/100
83/83          0s 865us/step -
accuracy: 0.8260 - loss: 0.3955
Epoch 60/100
83/83          0s 897us/step -
accuracy: 0.8185 - loss: 0.3939
Epoch 61/100
83/83          0s 889us/step -
accuracy: 0.8263 - loss: 0.3994
Epoch 62/100
83/83          0s 1ms/step -
accuracy: 0.8108 - loss: 0.4040
Epoch 63/100
```

```
83/83            0s 930us/step -
accuracy: 0.8249 - loss: 0.3823
Epoch 64/100
83/83            0s 973us/step -
accuracy: 0.8189 - loss: 0.3991
Epoch 65/100
83/83            0s 930us/step -
accuracy: 0.8250 - loss: 0.3846
Epoch 66/100
83/83            0s 952us/step -
accuracy: 0.8168 - loss: 0.3941
Epoch 67/100
83/83            0s 941us/step -
accuracy: 0.8163 - loss: 0.4002
Epoch 68/100
83/83            0s 939us/step -
accuracy: 0.8112 - loss: 0.3922
Epoch 69/100
83/83            0s 873us/step -
accuracy: 0.8219 - loss: 0.3894
Epoch 70/100
83/83            0s 865us/step -
accuracy: 0.8197 - loss: 0.3958
Epoch 71/100
83/83            0s 861us/step -
accuracy: 0.8255 - loss: 0.3796
Epoch 72/100
83/83            0s 881us/step -
accuracy: 0.8301 - loss: 0.3852
Epoch 73/100
83/83            0s 901us/step -
accuracy: 0.8214 - loss: 0.3841
Epoch 74/100
83/83            0s 852us/step -
accuracy: 0.8144 - loss: 0.3992
Epoch 75/100
83/83            0s 902us/step -
accuracy: 0.8122 - loss: 0.3939
Epoch 76/100
83/83            0s 2ms/step -
accuracy: 0.8291 - loss: 0.3719
Epoch 77/100
83/83            0s 866us/step -
accuracy: 0.8306 - loss: 0.3734
Epoch 78/100
83/83            0s 867us/step -
accuracy: 0.8258 - loss: 0.3680
Epoch 79/100
```

```
83/83            0s 942us/step -
accuracy: 0.8340 - loss: 0.3777
Epoch 80/100
83/83            0s 878us/step -
accuracy: 0.8244 - loss: 0.3760
Epoch 81/100
83/83            0s 869us/step -
accuracy: 0.8201 - loss: 0.4006
Epoch 82/100
83/83            0s 889us/step -
accuracy: 0.8322 - loss: 0.3666
Epoch 83/100
83/83            0s 873us/step -
accuracy: 0.8229 - loss: 0.3836
Epoch 84/100
83/83            0s 858us/step -
accuracy: 0.8281 - loss: 0.3851
Epoch 85/100
83/83            0s 872us/step -
accuracy: 0.8288 - loss: 0.3862
Epoch 86/100
83/83            0s 865us/step -
accuracy: 0.8407 - loss: 0.3648
Epoch 87/100
83/83            0s 864us/step -
accuracy: 0.8355 - loss: 0.3652
Epoch 88/100
83/83            0s 862us/step -
accuracy: 0.8332 - loss: 0.3856
Epoch 89/100
83/83            0s 1ms/step -
accuracy: 0.8323 - loss: 0.3641
Epoch 90/100
83/83            0s 969us/step -
accuracy: 0.8422 - loss: 0.3603
Epoch 91/100
83/83            0s 916us/step -
accuracy: 0.8334 - loss: 0.3661
Epoch 92/100
83/83            0s 949us/step -
accuracy: 0.8391 - loss: 0.3562
Epoch 93/100
83/83            0s 872us/step -
accuracy: 0.8276 - loss: 0.3862
Epoch 94/100
83/83            0s 851us/step -
accuracy: 0.8289 - loss: 0.3749
Epoch 95/100
```

```
83/83              0s 886us/step -
accuracy: 0.8247 - loss: 0.3805
Epoch 96/100
83/83              0s 867us/step -
accuracy: 0.8516 - loss: 0.3609
Epoch 97/100
83/83              0s 940us/step -
accuracy: 0.8325 - loss: 0.3615
Epoch 98/100
83/83              0s 890us/step -
accuracy: 0.8414 - loss: 0.3593
Epoch 99/100
83/83              0s 1ms/step -
accuracy: 0.8315 - loss: 0.3656
Epoch 100/100
83/83              0s 1ms/step -
accuracy: 0.8372 - loss: 0.3692
44/44              0s 1ms/step -
accuracy: 0.6809 - loss: 0.6817
[0.6950002312660217, 0.6680881381034851]
44/44              0s 2ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.61      0.73      1033
           1       0.43      0.82      0.57       374

    accuracy                           0.67      1407
   macro avg       0.67      0.72      0.65      1407
weighted avg       0.78      0.67      0.69      1407
```

[131]:
```python
y_pred_final=y_pred1.copy()
for i in range(len(y_pred1)):
    n_ones=y_pred1[i]+y_pred2[i]+y_pred3[i]
    if n_ones>1:
        y_pred_final[i]=1
    else:
        y_pred_final[i]=0
```

[132]:
```python
print(classification_report(y_test,y_pred_final))
```

```
             precision    recall  f1-score   support

           0       0.91      0.68      0.78      1033
           1       0.48      0.81      0.60       374

    accuracy                           0.71      1407
```

```
   macro avg       0.69      0.74      0.69      1407
weighted avg       0.79      0.71      0.73      1407
```

```
[ ]:
```