churn deep learning

May 11, 2023

[137]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

[138]:
```
df=pd.read_csv(r"C:\Users\Rakesh\Downloads\archive\churn.csv")
```

[139]:
```
df.head()
```

[139]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	…	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
0	7590-VHVEG	Female	0	Yes	No	1	No	…	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	No
1	5575-GNVDE	Male	0	No	No	34	Yes	…	Yes	No	No	No	One year	No	Mailed check	56.95	1889.5	No
2	3668-QPYBK	Male	0	No	No	2	Yes	…	No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	Yes
3	7795-CFOCW	Male	0	No	No	45	No	…	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	No
4	9237-HQITU	Female	0	No	No	2	Yes	…	No	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	Yes

[5 rows x 21 columns]

[140]:
```
df.drop('customerID',axis='columns',inplace=True)
```

[141]:
```
df.dtypes
```

[141]:
```
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
```

[142]:
```
df.TotalCharges.values
```

[142]:
```
array(['29.85', '1889.5', '108.15', …, '346.45', '306.6', '6844.5'],
      dtype=object)
```

[143]:
```
pd.to_numeric(df.TotalCharges)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
File ~\anaconda3\lib\site-packages\pandas\_libs\lib.pyx:2315, 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)
Input In [143], in <cell line: 1>()
----> 1 pd.to_numeric(df.TotalCharges)

File ~\anaconda3\lib\site-packages\pandas\core\tools\numeric.py:184, in␣
 ↪to_numeric(arg, errors, downcast)
```

```
    182 coerce_numeric = errors not in ("ignore", "raise")
    183 try:
--> 184     values, _ = lib.maybe_convert_numeric(
    185         values, set(), coerce_numeric=coerce_numeric
    186     )
    187 except (ValueError, TypeError):
    188     if errors == "raise":

File ~\anaconda3\lib\site-packages\pandas\_libs\lib.pyx:2357, in pandas._libs.
 ↪lib.maybe_convert_numeric()

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

[144]:
```
pd.to_numeric(df.TotalCharges).isnull()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
File ~\anaconda3\lib\site-packages\pandas\_libs\lib.pyx:2315, 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)
Input In [144], in <cell line: 1>()
----> 1 pd.to_numeric(df.TotalCharges).isnull()

File ~\anaconda3\lib\site-packages\pandas\core\tools\numeric.py:184, in␣
 ↪to_numeric(arg, errors, downcast)
    182 coerce_numeric = errors not in ("ignore", "raise")
    183 try:
--> 184     values, _ = lib.maybe_convert_numeric(
    185         values, set(), coerce_numeric=coerce_numeric
    186     )
    187 except (ValueError, TypeError):
    188     if errors == "raise":

File ~\anaconda3\lib\site-packages\pandas\_libs\lib.pyx:2357, in pandas._libs.
 ↪lib.maybe_convert_numeric()

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

[145]:
```
df.iloc[488]
```

[145]:
```
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
```

[146]:
```
df[pd.to_numeric(df.TotalCharges,errors='coerce').isnull()].shape
```

[146]:
```
(11, 20)
```

[147]:
```
df1=df.drop[pd.to_numeric(df.TotalCharges,errors='coerce').isnull()]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [147], in <cell line: 1>()
----> 1 df1=df.drop[pd.to_numeric(df.TotalCharges,errors='coerce').isnull()]

TypeError: 'method' object is not subscriptable
```

[148]:
```
df1=df[df.TotalCharges!=" "]
```

[149]:
```
df1['TotalCharges']=pd.to_numeric(df1.TotalCharges)
```

```
C:\Users\Rakesh\AppData\Local\Temp\ipykernel_17716\3081713981.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)
```

[150]:
```
df1['TotalCharges'].dtypes
```

[150]:
```
dtype('float64')
```

[151]:
```
df1.dtypes
```

[151]:
```
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
```

[152]:
```
df1['gender'].replace({'Female':0,'Male':1},inplace=True)
```

```
C:\Users\Rakesh\AppData\Local\Temp\ipykernel_17716\4018057419.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':0,'Male':1},inplace=True)
```

[153]:
```
df1['gender']
```

[153]:
```
0       0
1       1
2       1
3       1
4       0
       ..
7038    1
7039    0
```

```
7040    0
7041    1
7042    1
Name: gender, Length: 7032, dtype: int64
```

[154]:
```
def Print_unique_value(df):
    for column in df:
        if df[column].dtype== object:
            print(f'{column}:{df[column].unique()}')
```

[155]:
```
Print_unique_value(df1)
```

```
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']
```

[156]:
```
df1.replace({'No phone service','No internet service'},'No',inplace=True)
```

```
C:\Users\Rakesh\AppData\Local\Temp\ipykernel_17716\2910884998.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 internet service'},'No',inplace=True)
```

[157]:
```
Print_unique_value(df1)
```

```
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']
```

[158]:
```
yes_no_columns=['Partner','Dependents','PhoneService','MultipleLines','OnlineSecurity','Online
                ␣
 ↪'DeviceProtection','TechSupport','StreamingTV','StreamingMovies','PaperlessBilling','Churn'
for col in yes_no_columns:
    df1[col].replace({'Yes':1,'No':0},inplace=True)
```

```
C:\Users\Rakesh\AppData\Local\Temp\ipykernel_17716\3562970964.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)
```

[162]:
```
Print_unique_value(df1)
```

```
InternetService:['DSL' 'Fiber optic' 'No']
Contract:['Month-to-month' 'One year' 'Two year']
PaymentMethod:['Electronic check' 'Mailed check' 'Bank transfer (automatic)'
 'Credit card (automatic)']
```

[163]:
```
df2=pd.
 ↪get_dummies(data=df1,columns=['InternetService','Contract','PaymentMethod'])
```

[164]:
```
Print_unique_value(df2)
```

[165]:
```
df2.head()
```

[165]:

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

   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

   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

   PaymentMethod_Bank transfer (automatic)  \
0                                        0
1                                        0
2                                        0
3                                        1
4                                        0

   PaymentMethod_Credit card (automatic)  PaymentMethod_Electronic check  \
0                                      0                                1
1                                      0                                0
2                                      0                                0
3                                      0                                0
4                                      0                                1

   PaymentMethod_Mailed check
0                           0
1                           1
2                           1
3                           0
4                           0

[5 rows x 27 columns]
```

[166]:
```
df2.dtypes
```

[166]:
```
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                          uint8
InternetService_Fiber optic                  uint8
InternetService_No                           uint8
Contract_Month-to-month                      uint8
Contract_One year                            uint8
Contract_Two year                            uint8
PaymentMethod_Bank transfer (automatic)      uint8
PaymentMethod_Credit card (automatic)        uint8
PaymentMethod_Electronic check               uint8
PaymentMethod_Mailed check                   uint8
dtype: object
```

[167]:
```
col_to_scale=['tenure','MonthlyCharges','TotalCharges']
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df2[col_to_scale] = scaler.fit_transform(df2[col_to_scale])
```

[168]:
```
for col in df2:
    print(f'{col}: {df2[col].unique()}')
```

```
gender: [0 1]
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]
```

[169]:
```
x=df2.drop('Churn',axis='columns')
y=df2['Churn']
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)
```

[170]:
```
x_train.shape
```

[170]:
```
(5625, 26)
```

[173]:
```
import tensorflow as tf
from tensorflow import keras
```

[175]:
```
model=keras.Sequential([
    keras.layers.Dense(26,input_shape=(26,),activation="relu"),
    keras.layers.Dense(15, 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
```

```
176/176 [==============================] - 2s 3ms/step - loss: 0.5334 -
accuracy: 0.7232
Epoch 2/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4299 -
accuracy: 0.7952
Epoch 3/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4208 -
accuracy: 0.7982
Epoch 4/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4147 -
accuracy: 0.8036
Epoch 5/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4117 -
accuracy: 0.8082
Epoch 6/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4095 -
accuracy: 0.8064
Epoch 7/100
176/176 [==============================] - 1s 4ms/step - loss: 0.4072 -
accuracy: 0.8128
Epoch 8/100
176/176 [==============================] - 1s 4ms/step - loss: 0.4065 -
accuracy: 0.8130
Epoch 9/100
176/176 [==============================] - 1s 4ms/step - loss: 0.4052 -
accuracy: 0.8126
Epoch 10/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4030 -
accuracy: 0.8121
Epoch 11/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4025 -
accuracy: 0.8144
Epoch 12/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3999 -
accuracy: 0.8158
Epoch 13/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3998 -
accuracy: 0.8165
Epoch 14/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3990 -
accuracy: 0.8130
Epoch 15/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3964 -
accuracy: 0.8140
Epoch 16/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3950 -
accuracy: 0.8167
Epoch 17/100
```

```
176/176 [==============================] - 1s 3ms/step - loss: 0.3946 -
accuracy: 0.8164
Epoch 18/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3934 -
accuracy: 0.8164
Epoch 19/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3921 -
accuracy: 0.8162
Epoch 20/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3911 -
accuracy: 0.8180
Epoch 21/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3910 -
accuracy: 0.8201
Epoch 22/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3898 -
accuracy: 0.8206
Epoch 23/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3883 -
accuracy: 0.8185
Epoch 24/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3882 -
accuracy: 0.8231
Epoch 25/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3868 -
accuracy: 0.8183
Epoch 26/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3860 -
accuracy: 0.8231
Epoch 27/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3847 -
accuracy: 0.8217
Epoch 28/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3840 -
accuracy: 0.8222
Epoch 29/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3834 -
accuracy: 0.8249
Epoch 30/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3826 -
accuracy: 0.8206
Epoch 31/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3819 -
accuracy: 0.8238
Epoch 32/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3802 -
accuracy: 0.8254
Epoch 33/100
```

```
176/176 [==============================] - 1s 4ms/step - loss: 0.3800 -
accuracy: 0.8222
Epoch 34/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3790 -
accuracy: 0.8226
Epoch 35/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3772 -
accuracy: 0.8258
Epoch 36/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3780 -
accuracy: 0.8226
Epoch 37/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3763 -
accuracy: 0.8261
Epoch 38/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3755 -
accuracy: 0.8252
Epoch 39/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3742 -
accuracy: 0.8242
Epoch 40/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3753 -
accuracy: 0.8258
Epoch 41/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3733 -
accuracy: 0.8252
Epoch 42/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3745 -
accuracy: 0.8281
Epoch 43/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3726 -
accuracy: 0.8263
Epoch 44/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3722 -
accuracy: 0.8254
Epoch 45/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3717 -
accuracy: 0.8276
Epoch 46/100
176/176 [==============================] - 1s 5ms/step - loss: 0.3690 -
accuracy: 0.8284
Epoch 47/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3697 -
accuracy: 0.8254
Epoch 48/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3686 -
accuracy: 0.8263
Epoch 49/100
```

```
176/176 [==============================] - 1s 3ms/step - loss: 0.3686 -
accuracy: 0.8270
Epoch 50/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3674 -
accuracy: 0.8276
Epoch 51/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3665 -
accuracy: 0.8286
Epoch 52/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3659 -
accuracy: 0.8306
Epoch 53/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3660 -
accuracy: 0.8297
Epoch 54/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3647 -
accuracy: 0.8297
Epoch 55/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3647 -
accuracy: 0.8295
Epoch 56/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3643 -
accuracy: 0.8295
Epoch 57/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3645 -
accuracy: 0.8322
Epoch 58/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3635 -
accuracy: 0.8306
Epoch 59/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3632 -
accuracy: 0.8306
Epoch 60/100
176/176 [==============================] - 0s 3ms/step - loss: 0.3628 -
accuracy: 0.8306
Epoch 61/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3617 -
accuracy: 0.8295
Epoch 62/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3607 -
accuracy: 0.8308
Epoch 63/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3611 -
accuracy: 0.8286
Epoch 64/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3618 -
accuracy: 0.8299
Epoch 65/100
```

```
176/176 [==============================] - 1s 4ms/step - loss: 0.3620 -
accuracy: 0.8315
Epoch 66/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3617 -
accuracy: 0.8288
Epoch 67/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3597 -
accuracy: 0.8306
Epoch 68/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3593 -
accuracy: 0.8320
Epoch 69/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3601 -
accuracy: 0.8309
Epoch 70/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3572 -
accuracy: 0.8343
Epoch 71/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3559 -
accuracy: 0.8357
Epoch 72/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3572 -
accuracy: 0.8299
Epoch 73/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3567 -
accuracy: 0.8332
Epoch 74/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3568 -
accuracy: 0.8331
Epoch 75/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3554 -
accuracy: 0.8368
Epoch 76/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3554 -
accuracy: 0.8332
Epoch 77/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3552 -
accuracy: 0.8359
Epoch 78/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3540 -
accuracy: 0.8324
Epoch 79/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3540 -
accuracy: 0.8359
Epoch 80/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3541 -
accuracy: 0.8364
Epoch 81/100
```

```
176/176 [==============================] - 1s 3ms/step - loss: 0.3547 -
accuracy: 0.8354
Epoch 82/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3545 -
accuracy: 0.8352
Epoch 83/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3520 -
accuracy: 0.8366
Epoch 84/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3530 -
accuracy: 0.8348
Epoch 85/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3524 -
accuracy: 0.8372
Epoch 86/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3524 -
accuracy: 0.8352
Epoch 87/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3521 -
accuracy: 0.8363
Epoch 88/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3509 -
accuracy: 0.8356
Epoch 89/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3513 -
accuracy: 0.8350
Epoch 90/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3519 -
accuracy: 0.8359
Epoch 91/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3510 -
accuracy: 0.8338
Epoch 92/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3504 -
accuracy: 0.8370
Epoch 93/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3488 -
accuracy: 0.8366
Epoch 94/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3483 -
accuracy: 0.8368
Epoch 95/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3497 -
accuracy: 0.8379
Epoch 96/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3497 -
accuracy: 0.8348
Epoch 97/100
```

```
176/176 [==============================] - 1s 3ms/step - loss: 0.3478 -
accuracy: 0.8402
Epoch 98/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3484 -
accuracy: 0.8356
Epoch 99/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3479 -
accuracy: 0.8386
Epoch 100/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3493 -
accuracy: 0.8343
```

[175]:
```
<keras.callbacks.History at 0x1845161f3a0>
```

[176]:
```
model.evaluate(x_test,y_test)
```

```
44/44 [==============================] - 0s 3ms/step - loss: 0.4815 - accuracy:
0.7690
```

[176]:
```
[0.4814586639404297, 0.7690120935440063]
```

[177]:
```
yp=model.predict(x_test)
```

```
44/44 [==============================] - 0s 2ms/step
```

[178]:
```
yp[:5]
```

[178]:
```
array([[0.25519264],
       [0.61172974],
       [0.00810203],
       [0.8479525 ],
       [0.21663639]], dtype=float32)
```

[179]:
```
ypre=[]
for i in yp:
    if i>0.5:
        ypre.append(1)
    else:
        ypre.append(0)
```

[180]:
```
ypre[:5]
```

[180]:
```
[0, 1, 0, 1, 0]
```

[182]:
```
from sklearn.metrics import confusion_matrix , classification_report

print(classification_report(y_test,ypre))
```

```
              precision    recall  f1-score   support
```

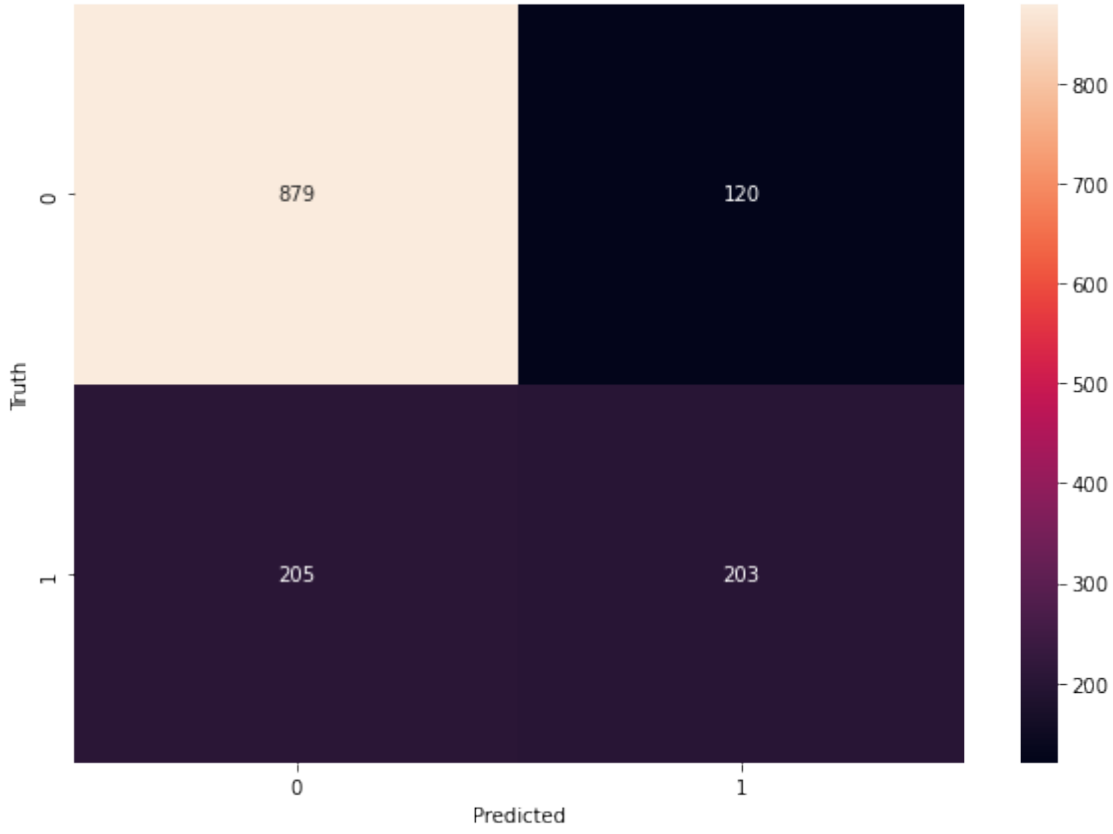
```
              0       0.81      0.88      0.84       999
              1       0.63      0.50      0.56       408

    accuracy                            0.77      1407
   macro avg       0.72      0.69      0.70      1407
weighted avg       0.76      0.77      0.76      1407
```

[183]:
```
import seaborn as sn
cm = tf.math.confusion_matrix(labels=y_test,predictions=ypre)

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

[183]: Text(69.0, 0.5, 'Truth')



[ ]: