churn practice

May 11, 2023

[155]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

[170]:
```
df=pd.read_csv(r"C:\Users\Rakesh\Downloads\archive\churn.csv")
```

[171]:
```
df.head(3)
```

[171]:
```
   customerID  gender  SeniorCitizen Partner Dependents  tenure PhoneService  \
0  7590-VHVEG  Female              0     Yes         No       1           No
1  5575-GNVDE    Male              0      No         No      34          Yes
2  3668-QPYBK    Male              0      No         No       2          Yes

      MultipleLines InternetService OnlineSecurity  … DeviceProtection  \
0  No phone service             DSL             No  …               No
1                No             DSL            Yes  …              Yes
2                No             DSL            Yes  …               No

  TechSupport StreamingTV StreamingMovies        Contract PaperlessBilling  \
0          No          No              No  Month-to-month              Yes
1          No          No              No        One year               No
2          No          No              No  Month-to-month              Yes

      PaymentMethod MonthlyCharges  TotalCharges Churn
0  Electronic check          29.85         29.85    No
1      Mailed check          56.95        1889.5    No
2      Mailed check          53.85        108.15   Yes

[3 rows x 21 columns]
```

[172]:
```
df.drop('customerID',axis='columns',inplace=True)
```

[173]:
```
df.dtypes
```

[173]:
```
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
```

[174]:
```python
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

[175]:
```python
df.gender=le.fit_transform(df['gender'])
```

[178]:
```python
df.dtypes
```

[178]:
```
gender                int32
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
```

[179]:
```
df.TotalCharges.values
```

[179]:
```
array(['29.85', '1889.5', '108.15', …, '346.45', '306.6', '6844.5'],
      dtype=object)
```

[180]:
```
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 [180], 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
```

[182]:
```
df[df.TotalCharges.isnull()].shape
```

[182]:
```
(0, 20)
```

[183]:
```
df1=df[df.TotalCharges!=" "]
```

[184]:
```
df1.dtypes
```

[184]:
```
gender                int32
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
```

[185]:
```
df1['TotalCharges']=pd.to_numeric(df1.TotalCharges)
```

```
C:\Users\Rakesh\AppData\Local\Temp\ipykernel_8156\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)
```

[186]:
```
df1.dtypes
```

[186]:
```
gender                 int32
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
```

[194]:
```
def print_unique(df):
    for col in df:
        if df[col].dtypes==object:
            print(f'{col}:{df[col].unique()}')
```

[195]:
```
print_unique(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']
```

[196]:
```
df1.replace({'No phone service','No internet service'},'No',inplace=True)
```

```
C:\Users\Rakesh\AppData\Local\Temp\ipykernel_8156\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)
```

[197]:
```
print_unique(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']
```

[200]:
```
replace_yes_no=['Partner','Dependents','PhoneService','MultipleLines','OnlineSecurity','Online
                     'DeviceProtection','TechSupport','StreamingTV',
                   'StreamingMovies','PaperlessBilling','Churn']
for i in replace_yes_no:
    df1[i].replace({'Yes':1,'No':0},inplace=True)
```

```
C:\Users\Rakesh\AppData\Local\Temp\ipykernel_8156\3498429156.py:5:
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[i].replace({'Yes':1,'No':0},inplace=True)
```

[201]:
```
print_unique(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)']
```

[202]:
```
df2=pd.
 ↪get_dummies(data=df1,columns=['InternetService','Contract','PaymentMethod'])
```

[203]:
```
print_unique(df2)
```

[204]:
```
df2.head(2)
```

[204]:
```
   gender  SeniorCitizen  Partner  Dependents  tenure  PhoneService  \
0       0              0        1           0       1             0
1       1              0        0           0      34             1

   MultipleLines  OnlineSecurity  OnlineBackup  DeviceProtection  …  \
0              0               0             1                 0  …
1              0               1             0                 1  …

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

```
   Contract_Month-to-month  Contract_One year  Contract_Two year  \
0                        1                  0                  0
1                        0                  1                  0

   PaymentMethod_Bank transfer (automatic)  \
0                                        0
1                                        0

   PaymentMethod_Credit card (automatic)  PaymentMethod_Electronic check  \
0                                      0                               1
1                                      0                               0

   PaymentMethod_Mailed check
0                           0
1                           1

[2 rows x 27 columns]
```

[206]:

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

[207]:

```
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]
```

[209]:
```
X=df2.drop('Churn',axis='columns')
y=df2['Churn']
```

[213]:
```
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)
```

[214]:
```
X_train.shape
```

[214]:
```
(5625, 26)
```

[215]:
```
X_test.shape
```

[215]:
```
(1407, 26)
```

[216]:
```
import tensorflow as tf
from tensorflow import keras
```

[218]:
```
model=keras.Sequential([
    keras.layers.Dense(30,input_shape=(26,),activation='relu'),
    keras.layers.Dense(12,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 [==============================] - 1s 2ms/step - loss: 0.5122 -
accuracy: 0.7372
Epoch 2/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4363 -
accuracy: 0.7886
Epoch 3/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4252 -
accuracy: 0.7948
Epoch 4/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4189 -
accuracy: 0.8000
Epoch 5/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4150 -
accuracy: 0.8036
Epoch 6/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4117 -
accuracy: 0.8078
Epoch 7/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4087 -
accuracy: 0.8062
Epoch 8/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4065 -
accuracy: 0.8103
Epoch 9/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4052 -
accuracy: 0.8100
Epoch 10/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4033 -
accuracy: 0.8119
Epoch 11/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4011 -
accuracy: 0.8130
Epoch 12/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3993 -
accuracy: 0.8124
Epoch 13/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3979 -
accuracy: 0.8144
Epoch 14/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3964 -
accuracy: 0.8151
Epoch 15/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3954 -
accuracy: 0.8153
Epoch 16/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3942 -
accuracy: 0.8132
```

```
Epoch 17/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3922 -
accuracy: 0.8169
Epoch 18/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3913 -
accuracy: 0.8180
Epoch 19/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3900 -
accuracy: 0.8155
Epoch 20/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3887 -
accuracy: 0.8167
Epoch 21/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3882 -
accuracy: 0.8167
Epoch 22/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3867 -
accuracy: 0.8174
Epoch 23/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3865 -
accuracy: 0.8155
Epoch 24/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3850 -
accuracy: 0.8172
Epoch 25/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3829 -
accuracy: 0.8180
Epoch 26/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3828 -
accuracy: 0.8217
Epoch 27/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3818 -
accuracy: 0.8208
Epoch 28/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3811 -
accuracy: 0.8187
Epoch 29/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3793 -
accuracy: 0.8187
Epoch 30/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3789 -
accuracy: 0.8187
Epoch 31/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3793 -
accuracy: 0.8208
Epoch 32/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3768 -
accuracy: 0.8208
```

```
Epoch 33/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3764 -
accuracy: 0.8206
Epoch 34/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3759 -
accuracy: 0.8199
Epoch 35/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3759 -
accuracy: 0.8212
Epoch 36/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3743 -
accuracy: 0.8208
Epoch 37/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3732 -
accuracy: 0.8251
Epoch 38/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3720 -
accuracy: 0.8220
Epoch 39/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3717 -
accuracy: 0.8219
Epoch 40/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3717 -
accuracy: 0.8249
Epoch 41/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3704 -
accuracy: 0.8231
Epoch 42/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3710 -
accuracy: 0.8249
Epoch 43/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3697 -
accuracy: 0.8226
Epoch 44/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3681 -
accuracy: 0.8220
Epoch 45/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3672 -
accuracy: 0.8256
Epoch 46/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3675 -
accuracy: 0.8260
Epoch 47/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3667 -
accuracy: 0.8276
Epoch 48/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3654 -
accuracy: 0.8284
```

```
Epoch 49/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3647 -
accuracy: 0.8247
Epoch 50/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3644 -
accuracy: 0.8254
Epoch 51/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3639 -
accuracy: 0.8235
Epoch 52/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3630 -
accuracy: 0.8260
Epoch 53/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3626 -
accuracy: 0.8270
Epoch 54/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3618 -
accuracy: 0.8267
Epoch 55/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3610 -
accuracy: 0.8256
Epoch 56/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3611 -
accuracy: 0.8300
Epoch 57/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3600 -
accuracy: 0.8286
Epoch 58/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3605 -
accuracy: 0.8277
Epoch 59/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3592 -
accuracy: 0.8254
Epoch 60/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3576 -
accuracy: 0.8297
Epoch 61/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3579 -
accuracy: 0.8286
Epoch 62/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3577 -
accuracy: 0.8304
Epoch 63/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3581 -
accuracy: 0.8245
Epoch 64/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3559 -
accuracy: 0.8315
```

```
Epoch 65/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3557 -
accuracy: 0.8318
Epoch 66/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3549 -
accuracy: 0.8279
Epoch 67/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3549 -
accuracy: 0.8290
Epoch 68/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3550 -
accuracy: 0.8279
Epoch 69/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3551 -
accuracy: 0.8290
Epoch 70/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3535 -
accuracy: 0.8327
Epoch 71/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3537 -
accuracy: 0.8300
Epoch 72/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3511 -
accuracy: 0.8363
Epoch 73/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3522 -
accuracy: 0.8306
Epoch 74/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3505 -
accuracy: 0.8324
Epoch 75/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3501 -
accuracy: 0.8318
Epoch 76/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3502 -
accuracy: 0.8318
Epoch 77/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3505 -
accuracy: 0.8268
Epoch 78/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3491 -
accuracy: 0.8318
Epoch 79/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3484 -
accuracy: 0.8343
Epoch 80/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3487 -
accuracy: 0.8336
```

```
Epoch 81/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3463 -
accuracy: 0.8345
Epoch 82/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3450 -
accuracy: 0.8350
Epoch 83/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3466 -
accuracy: 0.8350
Epoch 84/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3455 -
accuracy: 0.8361
Epoch 85/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3448 -
accuracy: 0.8332
Epoch 86/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3443 -
accuracy: 0.8356
Epoch 87/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3453 -
accuracy: 0.8372
Epoch 88/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3430 -
accuracy: 0.8341
Epoch 89/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3444 -
accuracy: 0.8373
Epoch 90/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3432 -
accuracy: 0.8352
Epoch 91/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3424 -
accuracy: 0.8368
Epoch 92/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3415 -
accuracy: 0.8384
Epoch 93/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3389 -
accuracy: 0.8398
Epoch 94/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3405 -
accuracy: 0.8372
Epoch 95/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3407 -
accuracy: 0.8356
Epoch 96/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3393 -
accuracy: 0.8396
```

```
Epoch 97/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3394 -
accuracy: 0.8370
Epoch 98/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3386 -
accuracy: 0.8359
Epoch 99/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3377 -
accuracy: 0.8402
Epoch 100/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3383 -
accuracy: 0.8382
```

[218]:
```
<keras.callbacks.History at 0x2aafcffbdc0>
```

[219]:
```python
model.evaluate(X_test,y_test)
```

```
44/44 [==============================] - 0s 2ms/step - loss: 0.5266 - accuracy:
0.7598
```

[219]:
```
[0.5266087651252747, 0.759772539138794]
```

[220]:
```python
yp=model.predict(X_test)
```

```
44/44 [==============================] - 0s 1ms/step
```

[221]:
```python
yp[:5]
```

[221]:
```
array([[0.34145868],
       [0.9688415 ],
       [0.13586332],
       [0.02293821],
       [0.00927109]], dtype=float32)
```

[225]:
```python
ypre=[]
for i in yp:
    if i>0.5:
        ypre.append(1)
    else:
         ypre.append(0)
```

[227]:
```python
ypre[:7]
```

[227]:
```
[0, 1, 0, 0, 0, 0, 0]
```

[228]:
```
from sklearn.metrics import confusion_matrix , classification_report
print(classification_report(y_test,ypre))
```

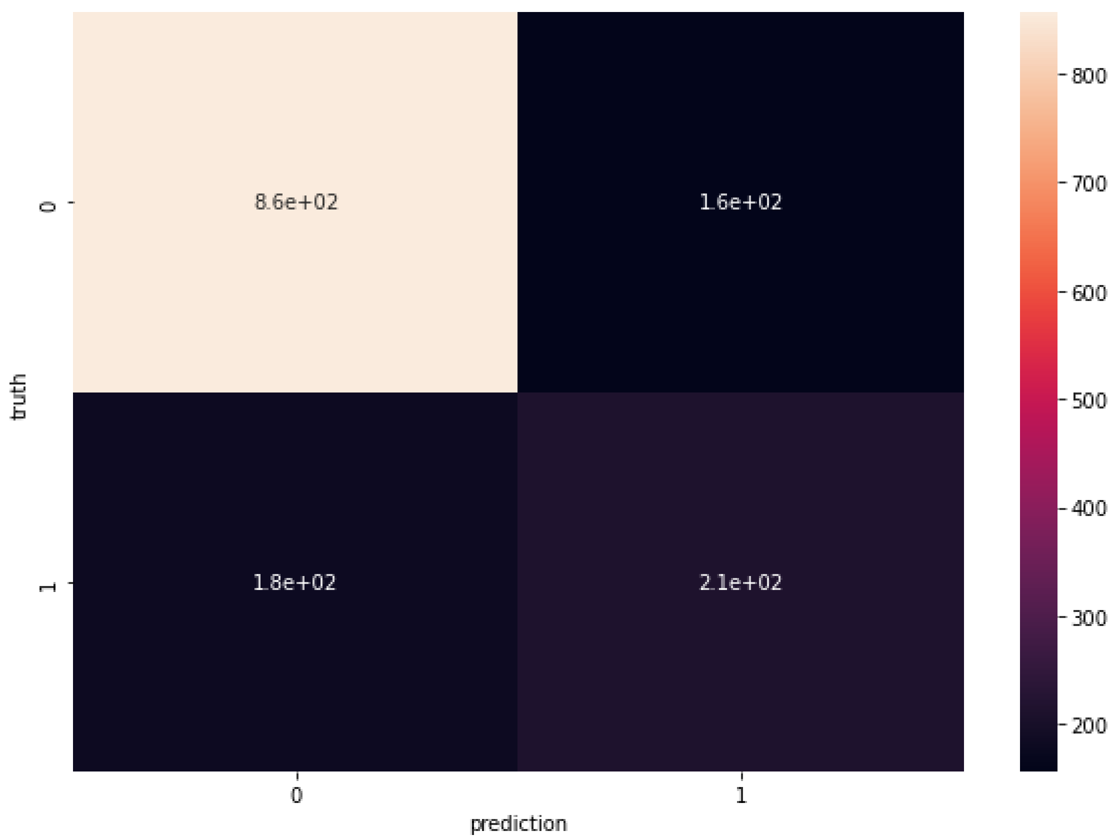
```
              precision    recall  f1-score   support

           0       0.83      0.85      0.84      1013
           1       0.58      0.54      0.56       394

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

[229]:
```
import seaborn as sn
cm=tf.math.confusion_matrix(labels=y_test,predictions=ypre)
plt.figure(figsize=(10,7))
sn.heatmap(cm,annot=True)
plt.xlabel('prediction')
plt.ylabel('truth')
```

[229]: Text(69.0, 0.5, 'truth')



```
[ ]:
```