Customer Churn Prediction Using Artificial Neural Network (ANN) __ Krutika Shinde

```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
%matplotlib inline

#Load the data

df = pd.read_csv("customer_churn.csv")
df.sample(5)
```

```
      customerID  gender  SeniorCitizen Partner Dependents  tenure  \
3021  6377-KSLXC    Male              0      No         No       5
4943  4973-MGTON  Female              0     Yes         No      71
4074  0946-CLJTI    Male              1     Yes         No      58
152   1679-JRFBR  Female              0     Yes        Yes      70
5044  9927-DSWDF    Male              0     Yes         No      22

     PhoneService MultipleLines InternetService
OnlineSecurity  ...  \
3021          Yes            No              No  No internet
service  ...
4943          Yes            No             DSL
Yes  ...
4074          Yes           Yes     Fiber optic
No  ...
152           Yes           Yes     Fiber optic
Yes  ...
5044          Yes            No     Fiber optic
Yes  ...

         DeviceProtection          TechSupport          StreamingTV  \
3021  No internet service  No internet service  No internet service
4943                  Yes                  Yes                  Yes
4074                  Yes                   No                  Yes
152                   Yes                   No                  Yes
5044                   No                  Yes                  Yes

          StreamingMovies        Contract PaperlessBilling  \
3021  No internet service  Month-to-month               No
4943                  Yes        Two year              Yes
4074                  Yes  Month-to-month              Yes
152                   Yes        One year              Yes
5044                  Yes  Month-to-month              Yes
```

```
                PaymentMethod MonthlyCharges  TotalCharges Churn
3021             Mailed check          19.95        107.05    No
4943  Credit card (automatic)          84.40        5969.3    No
4074         Electronic check          98.70        5812.6   Yes
152   Credit card (automatic)         108.15       7930.55    No
5044         Electronic check         104.60       2180.55    No

[5 rows x 21 columns]
```

```
#First of all, drop customerID column as it is of no use

df.drop('customerID',axis='columns',inplace=True)

df.dtypes
```

```
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
```

```
#Quick glance at above makes me realize that TotalCharges should be
float but it is an object.
#Let's check what's going on with this column

df.TotalCharges.values
```

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

pd.to_numeric(df.TotalCharges,errors='coerce').isnull()
```

```
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
```

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

```
      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
```

```
df.shape
```

```
(7043, 20)
```

```
df1 = df[df.TotalCharges!=' ']
df1.shape
```

```
(7032, 20)
```

```
#Remove rows with space in TotalCharges
```

```
df1.dtypes
```

```
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
```

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

```
C:\Users\kruti\AppData\Local\Temp\ipykernel_11424\973151263.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)
```

```
df1.TotalCharges.values
```

```
array([  29.85, 1889.5 ,  108.15, ...,  346.45,  306.6 , 6844.5 ])
```

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

```
      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
```

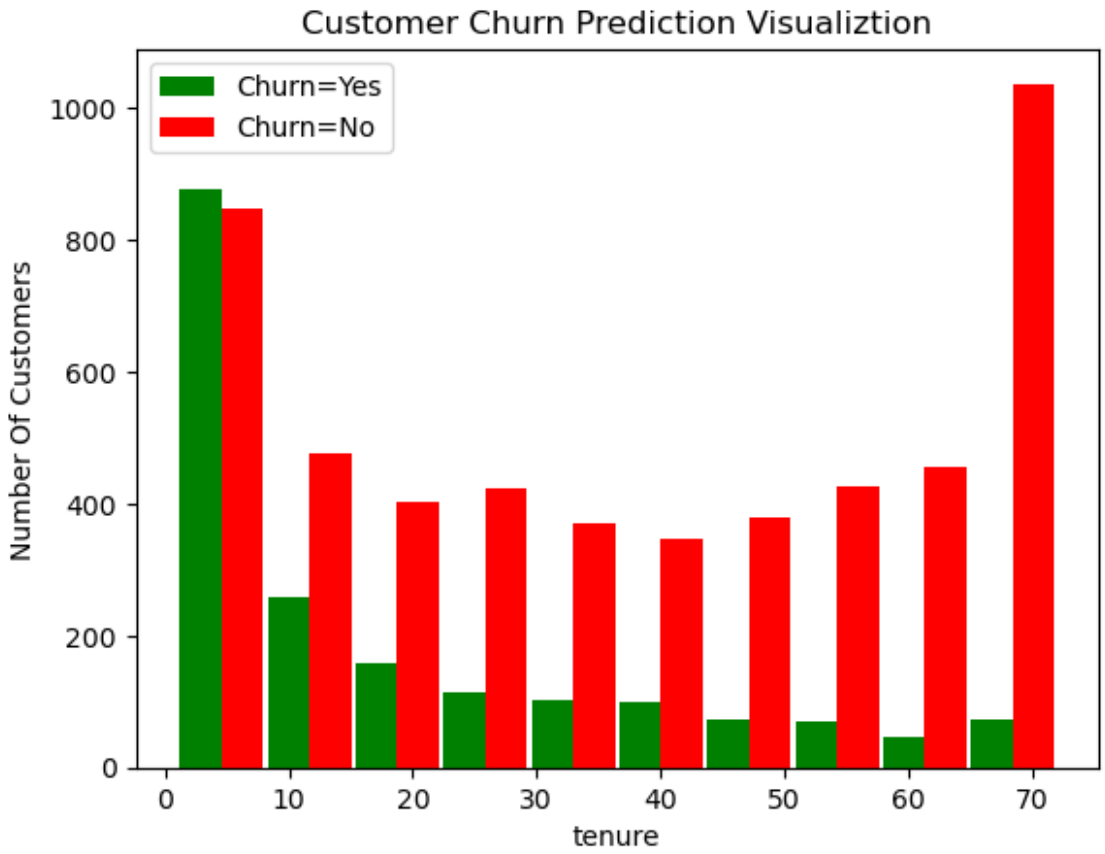
```
7042    Male              0      No         No      66          Yes

         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]
```

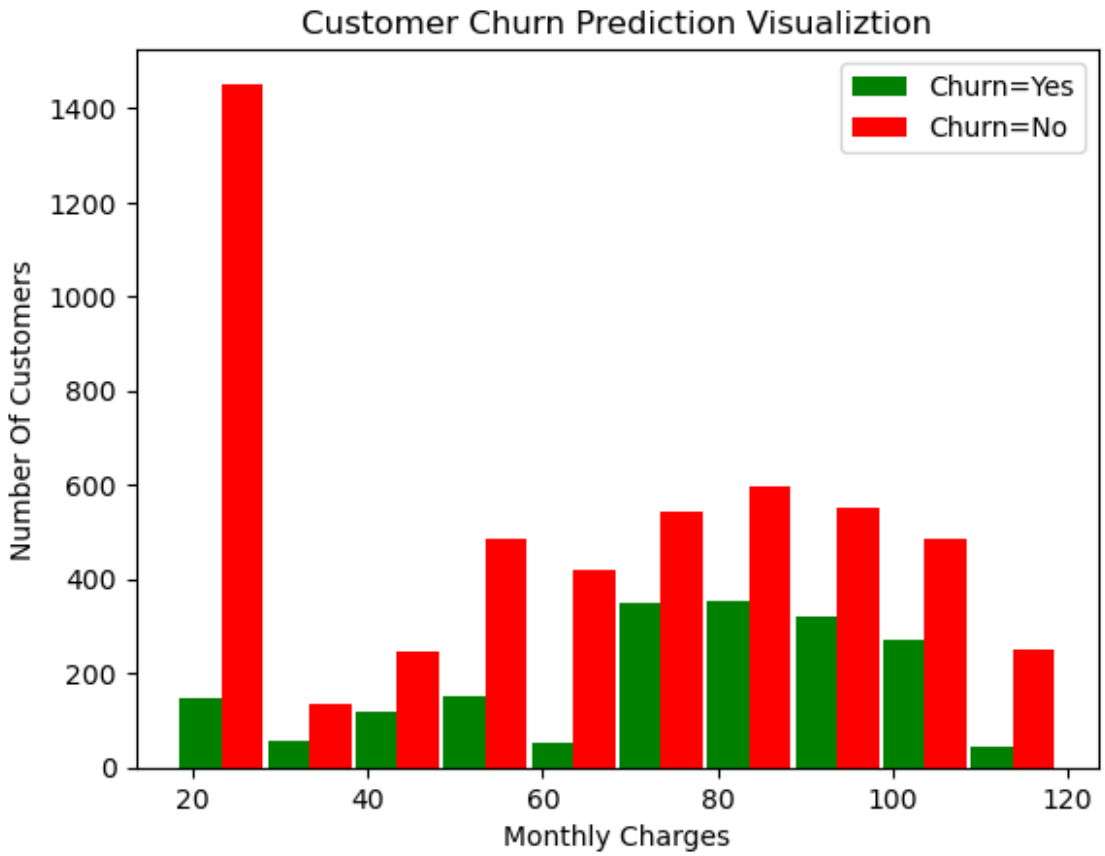
```
#Data Visualization

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

plt.xlabel("tenure")
plt.ylabel("Number Of Customers")
plt.title("Customer Churn Prediction Visualiztion")

blood_sugar_men = [113, 85, 90, 150, 149, 88, 93, 115, 135, 80, 77,
82, 129]
blood_sugar_women = [67, 98, 89, 120, 133, 150, 84, 69, 89, 79, 120,
112, 100]

plt.hist([tenure_churn_yes, tenure_churn_no], rwidth=0.95,
color=['green','red'],label=['Churn=Yes','Churn=No'])
plt.legend()
```

```
<matplotlib.legend.Legend at 0x29c76143670>
```




```
mc_churn_no = df1[df1.Churn=='No'].MonthlyCharges
mc_churn_yes = df1[df1.Churn=='Yes'].MonthlyCharges

plt.xlabel("Monthly Charges")
plt.ylabel("Number Of Customers")
plt.title("Customer Churn Prediction Visualiztion")

blood_sugar_men = [113, 85, 90, 150, 149, 88, 93, 115, 135, 80, 77,
82, 129]
blood_sugar_women = [67, 98, 89, 120, 133, 150, 84, 69, 89, 79, 120,
112, 100]

plt.hist([mc_churn_yes, mc_churn_no], rwidth=0.95,
color=['green','red'],label=['Churn=Yes','Churn=No'])
plt.legend()
```

```
<matplotlib.legend.Legend at 0x29c760bfac0>
```




```
#Many of the columns are yes, no etc. Let's print unique values in
object columns to see data values

def print_unique_col_values(df):
       for column in df:
            if df[column].dtypes=='object':
                print(f'{column}: {df[column].unique()}')

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']
```

```
#Some of the columns have no internet service or no phone service,
that can be replaced with a simple No
```

```
df1.replace('No internet service','No',inplace=True)
df1.replace('No phone service','No',inplace=True)
```

```
C:\Users\kruti\AppData\Local\Temp\ipykernel_11424\2045096646.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)
C:\Users\kruti\AppData\Local\Temp\ipykernel_11424\2045096646.py:2:
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)
```

```
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']
```

```
#Convert Yes and No to 1 or 0
```

```
yes_no_columns =
['Partner','Dependents','PhoneService','MultipleLines','OnlineSecurity
','OnlineBackup',

'DeviceProtection','TechSupport','StreamingTV','StreamingMovies','Pape
rlessBilling','Churn']
for col in yes_no_columns:
    df1[col].replace({'Yes': 1,'No': 0},inplace=True)
```

```
C:\Users\kruti\AppData\Local\Temp\ipykernel_11424\1648037665.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)
```

```
for col in df1:
    print(f'{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]
```

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

```
C:\Users\kruti\AppData\Local\Temp\ipykernel_11424\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)
```

```
df1.gender.unique()
```

```
array([1, 0], dtype=int64)
```

```
#One hot encoding for categorical columns
```

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

```
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')
```

```
df2.sample(5)
```

```
      gender  SeniorCitizen  Partner  Dependents  tenure  PhoneService
\
3246       0              0        0           0       3             1

22         0              0        0           0       1             1

4603       0              0        1           0       3             0

3522       0              0        1           1      37             1

1266       1              1        0           0       4             1


      MultipleLines  OnlineSecurity  OnlineBackup
DeviceProtection  ...  \
```

```
3246               0                0              0
0  ...
22                 0                0              0
0  ...
4603               0                1              0
0  ...
3522               0                0              0
0  ...
1266               1                1              0
0  ...

      InternetService_DSL  InternetService_Fiber optic
InternetService_No  \
3246                    1                            0
0
22                      0                            0
1
4603                    1                            0
0
3522                    0                            0
1
1266                    0                            1
0

      Contract_Month-to-month  Contract_One year  Contract_Two year  \
3246                        1                  0                  0
22                          1                  0                  0
4603                        1                  0                  0
3522                        0                  1                  0
1266                        1                  0                  0

      PaymentMethod_Bank transfer (automatic)  \
3246                                        0
22                                          0
4603                                        0
3522                                        0
1266                                        0

      PaymentMethod_Credit card (automatic)  PaymentMethod_Electronic
check  \
3246                                      0
0
22                                        0
0
4603                                      0
0
3522                                      1
0
1266                                      0
1
```

```
      PaymentMethod_Mailed check
3246                           1
22                             1
4603                           1
3522                           0
1266                           0

[5 rows x 27 columns]
```

```
df2.dtypes
```

```
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
```

```
cols_to_scale = ['tenure','MonthlyCharges','TotalCharges']

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df2[cols_to_scale] = scaler.fit_transform(df2[cols_to_scale])

for col in df2:
    print(f'{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]
```

```
#Train test split

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)
```

```
X_train.shape
```

```
(5625, 26)
```

```
X_test.shape
```

```
(1407, 26)
```

```
X_train[:10]
```

```
      gender  SeniorCitizen  Partner  Dependents    tenure
PhoneService  \
5664       1              1        0           0  0.126761
1
101        1              0        1           1  0.000000
1
2621       0              0        1           0  0.985915
1
392        1              1        0           0  0.014085
1
1327       0              0        1           0  0.816901
1
3607       1              0        0           0  0.169014
1
2773       0              0        1           0  0.323944
0
1936       1              0        1           0  0.704225
1
5387       0              0        0           0  0.042254
0
4331       0              0        0           0  0.985915
1

      MultipleLines  OnlineSecurity  OnlineBackup
DeviceProtection  ...  \
5664              0               0             0
1  ...
101               0               0             0
0  ...
2621              0               0             1
1  ...
392               0               0             0
0  ...
1327              1               0             0
1  ...
3607              0               1             0
0  ...
2773              0               0             0
1  ...
1936              0               1             1
```

```
0  ...
5387               0               0              0
0  ...
4331               1               0              0
0  ...

      InternetService_DSL  InternetService_Fiber optic
InternetService_No  \
5664                    0                            1
0
101                     0                            0
1
2621                    1                            0
0
392                     1                            0
0
1327                    0                            1
0
3607                    1                            0
0
2773                    1                            0
0
1936                    1                            0
0
5387                    1                            0
0
4331                    0                            0
1

      Contract_Month-to-month  Contract_One year  Contract_Two year  \
5664                        1                  0                  0
101                         1                  0                  0
2621                        0                  0                  1
392                         1                  0                  0
1327                        0                  1                  0
3607                        0                  1                  0
2773                        1                  0                  0
1936                        0                  1                  0
5387                        1                  0                  0
4331                        0                  0                  1

      PaymentMethod_Bank transfer (automatic)  \
5664                                        0
101                                         0
2621                                        0
392                                         0
1327                                        1
3607                                        0
2773                                        0
1936                                        1
```

```
5387                                                 0
4331                                                 1

      PaymentMethod_Credit card (automatic)  PaymentMethod_Electronic
check  \
5664                                                1
0
101                                                 0
1
2621                                                1
0
392                                                 0
1
1327                                                0
0
3607                                                0
0
2773                                                0
1
1936                                                0
0
5387                                                0
1
4331                                                0
0

      PaymentMethod_Mailed check
5664                            0
101                             0
2621                            0
392                             0
1327                            0
3607                            1
2773                            0
1936                            0
5387                            0
4331                            0

[10 rows x 26 columns]
```

```
len(X_train.columns)
```

```
26
```

```
#Build a model (ANN) in tensorflow/keras

import tensorflow as tf
from tensorflow import keras


model = keras.Sequential([
```

```
    keras.layers.Dense(26, input_shape=(26,), activation='relu'),
    keras.layers.Dense(15, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])

# opt = keras.optimizers.Adam(learning_rate=0.01)

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: 0.4923
- accuracy: 0.7509
Epoch 2/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4310
- accuracy: 0.7934
Epoch 3/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4225
- accuracy: 0.8014
Epoch 4/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4176
- accuracy: 0.8027
Epoch 5/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4141
- accuracy: 0.8073
Epoch 6/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4131
- accuracy: 0.8062
Epoch 7/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4112
- accuracy: 0.8075
Epoch 8/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4091
- accuracy: 0.8084
Epoch 9/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4077
- accuracy: 0.8094
Epoch 10/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4062
- accuracy: 0.8119
Epoch 11/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4056
- accuracy: 0.8117
Epoch 12/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4038
- accuracy: 0.8108
Epoch 13/100
```

```
176/176 [==============================] - 1s 3ms/step - loss: 0.4028
- accuracy: 0.8096
Epoch 14/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4015
- accuracy: 0.8103
Epoch 15/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4004
- accuracy: 0.8140
Epoch 16/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3986
- accuracy: 0.8146
Epoch 17/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3982
- accuracy: 0.8144
Epoch 18/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3973
- accuracy: 0.8148
Epoch 19/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3968
- accuracy: 0.8144
Epoch 20/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3958
- accuracy: 0.8139
Epoch 21/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3935
- accuracy: 0.8171
Epoch 22/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3941
- accuracy: 0.8153
Epoch 23/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3920
- accuracy: 0.8165
Epoch 24/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3913
- accuracy: 0.8149
Epoch 25/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3909
- accuracy: 0.8142
Epoch 26/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3906
- accuracy: 0.8156
Epoch 27/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3892
- accuracy: 0.8171
Epoch 28/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3889
- accuracy: 0.8133
Epoch 29/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3872
```

```
- accuracy: 0.8190
Epoch 30/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3868
- accuracy: 0.8160
Epoch 31/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3862
- accuracy: 0.8180
Epoch 32/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3850
- accuracy: 0.8178
Epoch 33/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3852
- accuracy: 0.8192
Epoch 34/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3830
- accuracy: 0.8203
Epoch 35/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3829
- accuracy: 0.8196
Epoch 36/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3833
- accuracy: 0.8176
Epoch 37/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3800
- accuracy: 0.8196
Epoch 38/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3821
- accuracy: 0.8187
Epoch 39/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3804
- accuracy: 0.8201
Epoch 40/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3794
- accuracy: 0.8185
Epoch 41/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3802
- accuracy: 0.8206
Epoch 42/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3786
- accuracy: 0.8204
Epoch 43/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3775
- accuracy: 0.8215
Epoch 44/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3774
- accuracy: 0.8192
Epoch 45/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3776
- accuracy: 0.8210
```

```
Epoch 46/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3750
- accuracy: 0.8201
Epoch 47/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3767
- accuracy: 0.8213
Epoch 48/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3746
- accuracy: 0.8212
Epoch 49/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3753
- accuracy: 0.8233
Epoch 50/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3737
- accuracy: 0.8263
Epoch 51/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3734
- accuracy: 0.8204
Epoch 52/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3722
- accuracy: 0.8219
Epoch 53/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3731
- accuracy: 0.8228
Epoch 54/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3717
- accuracy: 0.8231
Epoch 55/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3718
- accuracy: 0.8236
Epoch 56/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3717
- accuracy: 0.8235
Epoch 57/100
176/176 [==============================] - 1s 4ms/step - loss: 0.3705
- accuracy: 0.8235
Epoch 58/100
176/176 [==============================] - 1s 7ms/step - loss: 0.3704
- accuracy: 0.8228
Epoch 59/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3685
- accuracy: 0.8244
Epoch 60/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3683
- accuracy: 0.8252
Epoch 61/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3667
- accuracy: 0.8263
Epoch 62/100
```

```
176/176 [==============================] - 0s 1ms/step - loss: 0.3693
- accuracy: 0.8244
Epoch 63/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3673
- accuracy: 0.8245
Epoch 64/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3662
- accuracy: 0.8261
Epoch 65/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3658
- accuracy: 0.8233
Epoch 66/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3656
- accuracy: 0.8238
Epoch 67/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3653
- accuracy: 0.8258
Epoch 68/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3640
- accuracy: 0.8274
Epoch 69/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3647
- accuracy: 0.8265
Epoch 70/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3641
- accuracy: 0.8304
Epoch 71/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3638
- accuracy: 0.8281
Epoch 72/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3659
- accuracy: 0.8254
Epoch 73/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3611
- accuracy: 0.8292
Epoch 74/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3626
- accuracy: 0.8263
Epoch 75/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3620
- accuracy: 0.8263
Epoch 76/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3606
- accuracy: 0.8286
Epoch 77/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3604
- accuracy: 0.8297
Epoch 78/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3609
```

```
- accuracy: 0.8279
Epoch 79/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3601
- accuracy: 0.8284
Epoch 80/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3608
- accuracy: 0.8290
Epoch 81/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3593
- accuracy: 0.8299
Epoch 82/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3595
- accuracy: 0.8299
Epoch 83/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3583
- accuracy: 0.8322
Epoch 84/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3583
- accuracy: 0.8300
Epoch 85/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3568
- accuracy: 0.8304
Epoch 86/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3565
- accuracy: 0.8306
Epoch 87/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3564
- accuracy: 0.8320
Epoch 88/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3557
- accuracy: 0.8327
Epoch 89/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3560
- accuracy: 0.8324
Epoch 90/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3560
- accuracy: 0.8300
Epoch 91/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3544
- accuracy: 0.8325
Epoch 92/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3546
- accuracy: 0.8322
Epoch 93/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3555
- accuracy: 0.8347
Epoch 94/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3534
- accuracy: 0.8318
```

```
Epoch 95/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3525
- accuracy: 0.8338
Epoch 96/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3526
- accuracy: 0.8331
Epoch 97/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3525
- accuracy: 0.8318
Epoch 98/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3522
- accuracy: 0.8334
Epoch 99/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3507
- accuracy: 0.8348
Epoch 100/100
176/176 [==============================] - 0s 1ms/step - loss: 0.3498
- accuracy: 0.8372

<keras.callbacks.History at 0x29c0572ceb0>

model.evaluate(X_test, y_test)

44/44 [==============================] - 0s 1ms/step - loss: 0.4779 -
accuracy: 0.7775

[0.4779191017150879, 0.7775408625602722]

yp = model.predict(X_test)
yp[:5]

44/44 [==============================] - 0s 969us/step

array([[0.2952634 ],
       [0.6559457 ],
       [0.03176874],
       [0.74946564],
       [0.31923044]], dtype=float32)

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

y_pred[:10]

[0, 1, 0, 1, 0, 1, 0, 1, 0, 0]

y_test[:10]
```

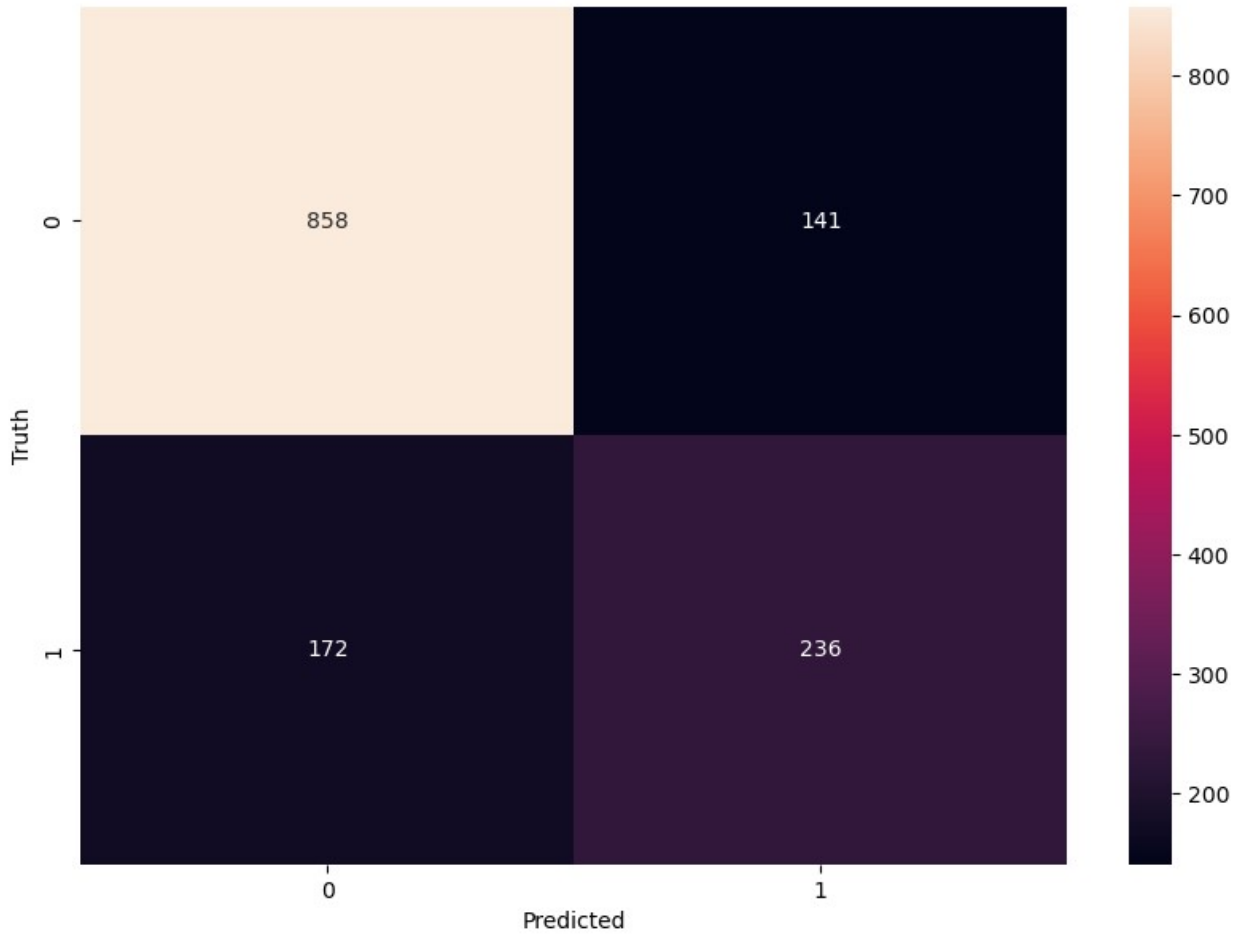
```
2660    0
744     0
5579    1
64      1
3287    1
816     1
2670    0
5920    0
1023    0
6087    0
Name: Churn, dtype: int64
```

```
from sklearn.metrics import confusion_matrix , classification_report

print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.86      0.85       999
           1       0.63      0.58      0.60       408

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

```
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')
```

```
Text(95.72222222222221, 0.5, 'Truth')
```




```
y_test.shape
```

```
(1407,)
```

```
#Accuracy
round((862+229)/(862+229+137+179),2)
```

```
0.78
```

```
#Precision for 0 class. i.e. Precision for customers who did not churn
round(862/(862+179),2)
```

```
0.83
```

```
#Precision for 1 class. i.e. Precision for customers who actually churned
round(229/(229+137),2)
```

```
0.63
```

```
#Recall for 0 class
round(862/(862+137),2)
```

```
0.86
```

```
round(229/(229+179),2)
```

```
0.56
```