In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [5]:
```
file_path = r"E:\ML\churn\WA_Fn-UseC_-Telco-Customer-Churn.csv"
df = pd.read_csv(file_path)
df.sample(5)
```

Out[5]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService
5933	6496-SLWHQ	Male	1	No	No	3	Yes
3497	9799-CAYJJ	Female	1	Yes	No	2	Yes
4377	8212-CRQXP	Female	0	Yes	No	22	Yes
6487	5998-DZLYR	Female	0	Yes	No	61	Yes
4188	2357-COQEK	Female	1	No	No	28	Yes

5 rows × 21 columns

In [237…
```
df.columns
```

Out[237…
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

In [239…
```
df.drop('customerID',axis='columns',inplace=True)
df.dtypes
```

```
Out[239…  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
```

```
In [241…  df.TotalCharges.values
```

```
Out[241…  array(['29.85', '1889.5', '108.15', ..., '346.45', '306.6', '6844.5'],
                dtype=object)
```

```
In [243…  pd.to_numeric(df.TotalCharges,errors='coerce').isnull()
```

```
Out[243…  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
```

```
In [245…  df[pd.to_numeric(df.TotalCharges,errors='coerce').isnull()]
```

Out[245…

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
488	Female	0	Yes	Yes	0	No	No phone service
753	Male	0	No	Yes	0	Yes	No
936	Female	0	Yes	Yes	0	Yes	No
1082	Male	0	Yes	Yes	0	Yes	Yes
1340	Female	0	Yes	Yes	0	No	No phone service
3331	Male	0	Yes	Yes	0	Yes	No
3826	Male	0	Yes	Yes	0	Yes	Yes
4380	Female	0	Yes	Yes	0	Yes	No
5218	Male	0	Yes	Yes	0	Yes	No
6670	Female	0	Yes	Yes	0	Yes	Yes
6754	Male	0	No	Yes	0	Yes	Yes

In [247…
```
df[pd.to_numeric(df.TotalCharges,errors='coerce').isnull()].shape
```
Out[247…
```
(11, 20)
```
In [249…
```
df.shape
```
Out[249…
```
(7043, 20)
```
In [251…
```
df1 = df[df.TotalCharges!=' ']
df1.shape
```
Out[251…
```
(7032, 20)
```
In [253…
```
df1.TotalCharges = pd.to_numeric(df1.TotalCharges)
```
```
C:\Users\kamalesh\AppData\Local\Temp\ipykernel_4796\973151263.py:1: SettingWithCo
pyWarning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1.TotalCharges = pd.to_numeric(df1.TotalCharges)
```
In [255…
```
df1.TotalCharges.dtypes
```

Out[255…

```
dtype('float64')
```

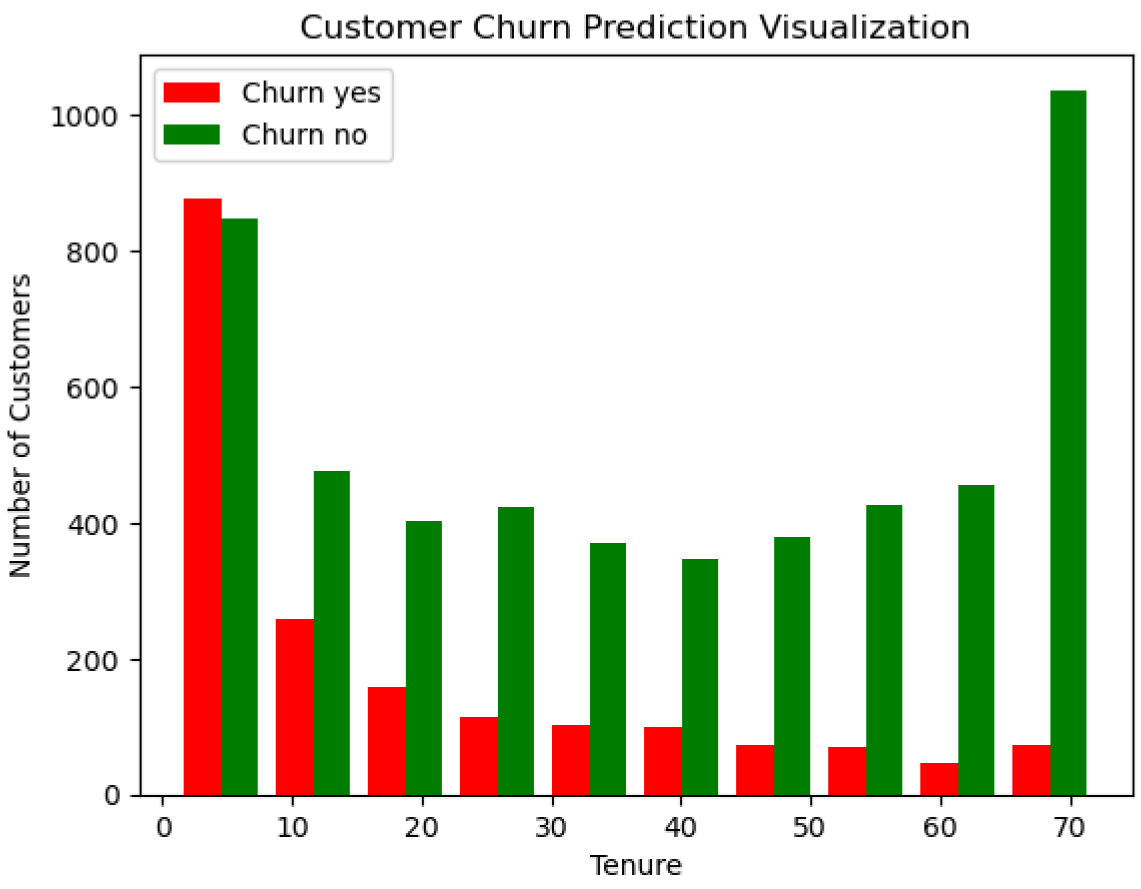
Data Visualization

In [258…

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

plt.hist([tenure_churn_yes,tenure_churn_no],color=['red','green'],label=['Churn
plt.xlabel('Tenure')
plt.ylabel('Number of Customers')
plt.title('Customer Churn Prediction Visualization')
plt.legend()
```

Out[258…

```
<matplotlib.legend.Legend at 0x278e4103410>
```

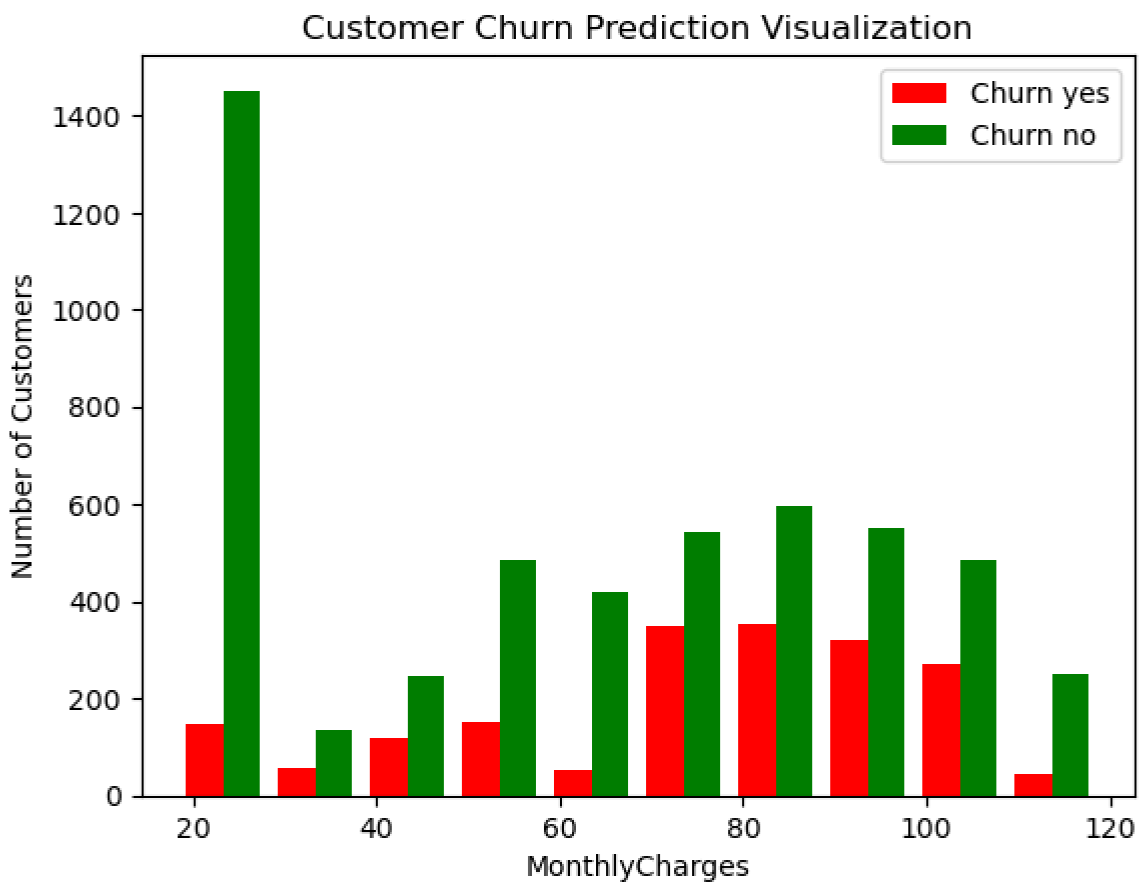


In [260…

```
MonthlyCharges_churn_no = df1[df1.Churn=='No'].MonthlyCharges
MonthlyCharges_churn_yes = df1[df1.Churn=='Yes'].MonthlyCharges

plt.hist([MonthlyCharges_churn_yes,MonthlyCharges_churn_no],color=['red','green'
plt.xlabel('MonthlyCharges')
plt.ylabel('Number of Customers')
plt.title('Customer Churn Prediction Visualization')
plt.legend()
```

Out[260…

```
<matplotlib.legend.Legend at 0x278e4352790>
```



In [262…

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

In [264…

```
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']
```

In [266…

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

```
C:\Users\kamalesh\AppData\Local\Temp\ipykernel_4796\2045096646.py:1: SettingWithC
opyWarning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1.replace('No internet service','No',inplace=True)
C:\Users\kamalesh\AppData\Local\Temp\ipykernel_4796\2045096646.py:2: SettingWithC
opyWarning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1.replace('No phone service','No',inplace=True)
```

In [268…

```
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']
```

In [270…

```
yes_no_columns = ['Partner','Dependents','PhoneService','MultipleLines','OnlineS
                  'StreamingMovies','PaperlessBilling','Churn']
for col in yes_no_columns:
    df1[col].replace({'Yes':1,'No':0},inplace=True)
```

```
C:\Users\kamalesh\AppData\Local\Temp\ipykernel_4796\2118069929.py:4: SettingWithC
opyWarning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1[col].replace({'Yes':1,'No':0},inplace=True)
```

In [272…

```
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]
```

In [274…
```
df1['gender'].replace({'Female':1,'Male':0},inplace=True)
```

```
C:\Users\kamalesh\AppData\Local\Temp\ipykernel_4796\698335744.py:1: SettingWithCo
pyWarning:
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/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['gender'].replace({'Female':1,'Male':0},inplace=True)
```

In [276…
```
df1['gender'].unique()
```

Out[276…
```
array([1, 0], dtype=int64)
```

In [278…
```
df2 = pd.get_dummies(data=df1,columns=['InternetService','Contract','PaymentMeth
```

In [280…
```
df2.head()
```

Out[280…

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	O
0	1	0	1	0	1	0	0	
1	0	0	0	0	34	1	0	
2	0	0	0	0	2	1	0	
3	0	0	0	0	45	0	0	
4	1	0	0	0	2	1	0	

5 rows × 27 columns

In [282…
```
df2.dtypes
```

```
Out[282…  gender                                     int32
          SeniorCitizen                              int32
          Partner                                    int32
          Dependents                                 int32
          tenure                                     int32
          PhoneService                               int32
          MultipleLines                              int32
          OnlineSecurity                             int32
          OnlineBackup                               int32
          DeviceProtection                           int32
          TechSupport                                int32
          StreamingTV                                int32
          StreamingMovies                            int32
          PaperlessBilling                           int32
          MonthlyCharges                             int32
          TotalCharges                               int32
          Churn                                      int32
          InternetService_DSL                        int32
          InternetService_Fiber optic                int32
          InternetService_No                         int32
          Contract_Month-to-month                    int32
          Contract_One year                          int32
          Contract_Two year                          int32
          PaymentMethod_Bank transfer (automatic)    int32
          PaymentMethod_Credit card (automatic)      int32
          PaymentMethod_Electronic check             int32
          PaymentMethod_Mailed check                 int32
          dtype: object
```

```
In [284…  cols_to_scale = ['tenure','MonthlyCharges','TotalCharges']

          from sklearn.preprocessing import MinMaxScaler
          scaler = MinMaxScaler()

          df2[cols_to_scale] = scaler.fit_transform(df2[cols_to_scale])
```

```
In [286…  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 = [0.11 0.38 0.35 0.24 0.52 0.81 0.71 0.86 0.31 0.   0.82 0.85 0.8
7 0.95
 0.02 0.88 0.37 0.72 0.21 0.01 0.41 0.12 0.46 0.78 0.77 0.48 0.27 0.51
 0.56 0.79 0.61 0.58 0.66 0.62 0.9  0.06 0.89 0.36 0.76 0.57 0.6  0.93
 0.92 0.07 0.32 0.44 0.15 0.64 0.03 0.8  0.39 0.73 0.13 0.67 0.7  0.05
 0.4  0.17 0.26 0.53 0.83 0.5  0.68 0.55 0.63 0.33 0.91 0.94 0.84 0.28
 0.75 0.47 0.3  0.22 0.65 0.97 0.96 0.23 0.43 0.42 0.49 0.54 0.69 0.74
 0.98 0.25 0.1  0.45 0.29 0.16 0.08 0.34 0.59 0.18 0.2  1.   0.99 0.04
 0.19]
TotalCharges = [0.00126933 0.21590122 0.01038541 ... 0.22755597 0.8474498  0.7876
7598]
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]
```

In [ ]:

```
```

In [289…

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

In [291…

```python
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=
```

In [293…

```python
X_train.shape
```

```
Out[293…  (5625, 26)
```

```
In [295…  X_test.shape
```

```
Out[295…  (1407, 26)
```

```
In [297…  X_train.head()
```

Out[297…

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLin
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	

5 rows × 26 columns

In [299…

```
import tensorflow as tf
from tensorflow import keras

model = keras.Sequential([
    keras.layers.Dense(20,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 = 50)
```

```
Epoch 1/50
C:\Users\kamalesh\AppData\Roaming\Python\Python311\site-packages\keras\src\layers
\core\dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` argument
to a layer. When using Sequential models, prefer using an `Input(shape)` object a
s the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
176/176 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.7062 - loss: 0.5589
Epoch 2/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7920 - loss: 0.4331
Epoch 3/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7921 - loss: 0.4369
Epoch 4/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8043 - loss: 0.4153
Epoch 5/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.7984 - loss: 0.4242
Epoch 6/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8103 - loss: 0.3955
Epoch 7/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8081 - loss: 0.4087
Epoch 8/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8053 - loss: 0.4187
Epoch 9/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8108 - loss: 0.3984
Epoch 10/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8116 - loss: 0.4061
Epoch 11/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8052 - loss: 0.4066
Epoch 12/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8152 - loss: 0.3911
Epoch 13/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8127 - loss: 0.4018
Epoch 14/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.7997 - loss: 0.4102
Epoch 15/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8160 - loss: 0.3879
Epoch 16/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8114 - loss: 0.4083
Epoch 17/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8188 - loss: 0.3924
Epoch 18/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8145 - loss: 0.3966
Epoch 19/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8157 - loss: 0.4055
Epoch 20/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8177 - loss: 0.3997
Epoch 21/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8140 - loss: 0.3927
Epoch 22/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8215 - loss: 0.3909
Epoch 23/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8297 - loss: 0.3883
Epoch 24/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8199 - loss: 0.3995
Epoch 25/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8247 - loss: 0.3901
Epoch 26/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8169 - loss: 0.3878
Epoch 27/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8166 - loss: 0.3991
Epoch 28/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8199 - loss: 0.3849
Epoch 29/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8200 - loss: 0.3897
Epoch 30/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8159 - loss: 0.3870
Epoch 31/50
```

```
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8158 - loss: 0.3938
Epoch 32/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8206 - loss: 0.3806
Epoch 33/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8136 - loss: 0.3900
Epoch 34/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8335 - loss: 0.3790
Epoch 35/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8262 - loss: 0.3791
Epoch 36/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8216 - loss: 0.3854
Epoch 37/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8177 - loss: 0.3928
Epoch 38/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8178 - loss: 0.3897
Epoch 39/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8137 - loss: 0.3994
Epoch 40/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8117 - loss: 0.3918
Epoch 41/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8184 - loss: 0.3978
Epoch 42/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8249 - loss: 0.3849
Epoch 43/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8180 - loss: 0.3966
Epoch 44/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8327 - loss: 0.3681
Epoch 45/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8196 - loss: 0.3843
Epoch 46/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8218 - loss: 0.3883
Epoch 47/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8284 - loss: 0.3760
Epoch 48/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.8195 - loss: 0.3817
Epoch 49/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8229 - loss: 0.3838
Epoch 50/50
176/176 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8168 - loss: 0.3924
```

Out[299…

```
<keras.src.callbacks.history.History at 0x278e42d2150>
```

In [300…

```
model.evaluate(X_test,y_test)
```

```
44/44 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8017 - loss: 0.4445
```

Out[300…

```
[0.4550953507423401, 0.7960199117660522]
```

In [303…

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

```
44/44 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
```

Out[303…

```
array([[0.22824892],
       [0.4995849 ],
       [0.00568537],
       [0.8144864 ],
       [0.6158154 ]], dtype=float32)
```

In [305…

```
y_test[:10]
```

```
Out[305…   2660    0
           744     0
           5579    1
           64      1
           3287    1
           816     1
           2670    0
           5920    0
           1023    0
           6087    0
           Name: Churn, dtype: int32
```

```
In [307…   y_pred = []
           for element in yp:
               if element > 0.5:
                   y_pred.append(1)
               else:
                   y_pred.append(0)
```

```
In [309…   y_pred[:10]
```

```
Out[309…   [0, 0, 0, 1, 1, 1, 0, 1, 0, 0]
```

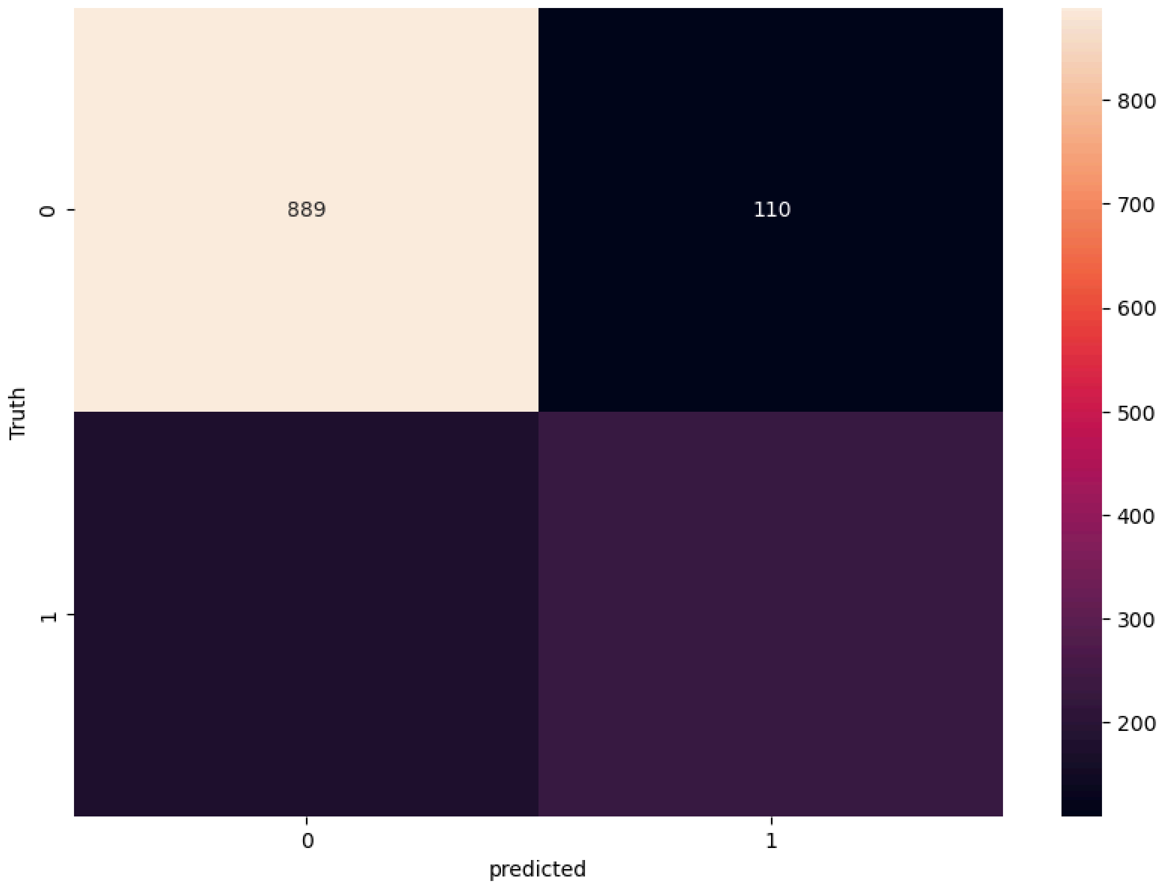
```
In [311…   from sklearn.metrics import confusion_matrix, classification_report
           print(classification_report(y_test,y_pred))
```

```
                 precision    recall  f1-score   support

              0       0.83      0.89      0.86       999
              1       0.68      0.57      0.62       408

       accuracy                           0.80      1407
      macro avg       0.76      0.73      0.74      1407
   weighted avg       0.79      0.80      0.79      1407
```

```
  In [ ]:
```

```
In [318…   import seaborn as sns
           cm = tf.math.confusion_matrix(labels=y_test,predictions=y_pred)

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

```
Out[318…   Text(95.72222222222221, 0.5, 'Truth')
```



```
In [ ]:
```