In [13]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

In [17]:
```
df=pd.read_csv('Downloads/WA_Fn-UseC_-Telco-Customer-Churn.csv')
df.sample(5)
```

Out[17]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity
5008	9150-KPBJQ	Female	0	No	No	6	Yes	No	No	No internet service
4011	8143-ETQTI	Female	0	Yes	Yes	23	Yes	No	No	No internet service
2431	8263-JQAIK	Male	1	No	No	2	No	No phone service	DSL	No
2185	7619-ODSGN	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes
3621	2933-FILNV	Female	0	Yes	Yes	2	No	No phone service	DSL	No

5 rows × 21 columns

In [18]:
```
df.drop('customerID', axis='columns',inplace=True)
df.dtypes
```

Out[18]:
```
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 [19]:
```
df.TotalCharges.values
```

Out[19]:
```
array(['29.85', '1889.5', '108.15', ..., '346.45', '306.6', '6844.5'],
      dtype=object)
```

In [20]:
```
df.MonthlyCharges.values
```

Out[20]:
```
array([ 29.85,  56.95,  53.85, ...,  29.6 ,  74.4 , 105.65])
```

In [24]:
```
pd.to_numeric(df.TotalCharges,errors='coerce').isnull()
```

Out[24]:
```
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 [31]:
```
df[pd.to_numeric(df.TotalCharges,errors='coerce').isnull()]
```

Out[31]:

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

In [27]:
```
df.shape
```

Out[27]:
```
(7043, 20)
```

In [29]:
```
df1=df[df.TotalCharges!=' ']
df1.shape
```

Out[29]:
```
(7032, 20)
```

In [33]:
```
df.iloc[488]['TotalCharges']
```

Out[33]:
```
' '
```

In [35]:
```
df1.shape
```

Out[35]:
```
(7032, 20)
```

In [36]:
```
df1.dtypes
```

Out[36]:
```
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 [38]:
```
df1.TotalCharges=pd.to_numeric(df1.TotalCharges)
```

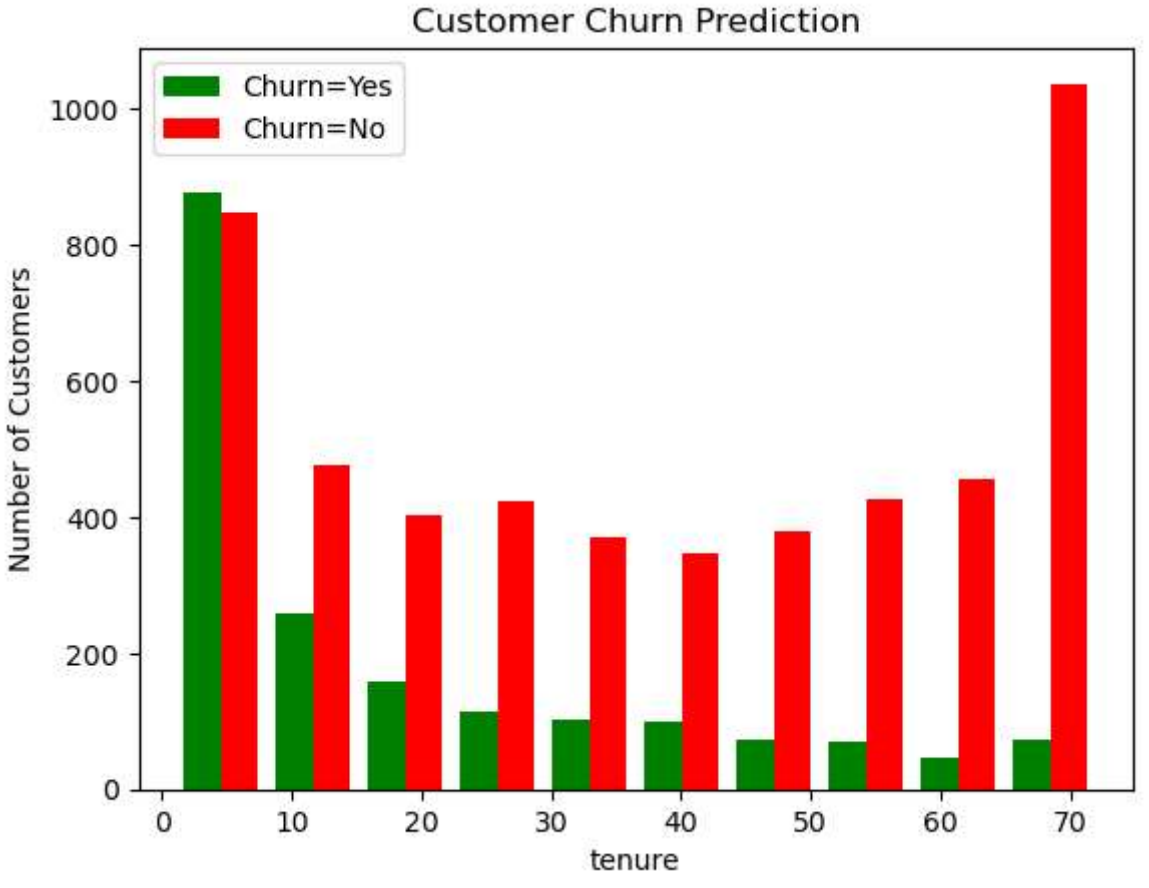
```
C:\Users\KIRAN\AppData\Local\Temp\ipykernel_17080\695980592.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#retu
rning-a-view-versus-a-copy
  df1.TotalCharges=pd.to_numeric(df1.TotalCharges)
```

In [39]:
```
df1.TotalCharges.dtypes
```

Out[39]:
```
dtype('float64')
```

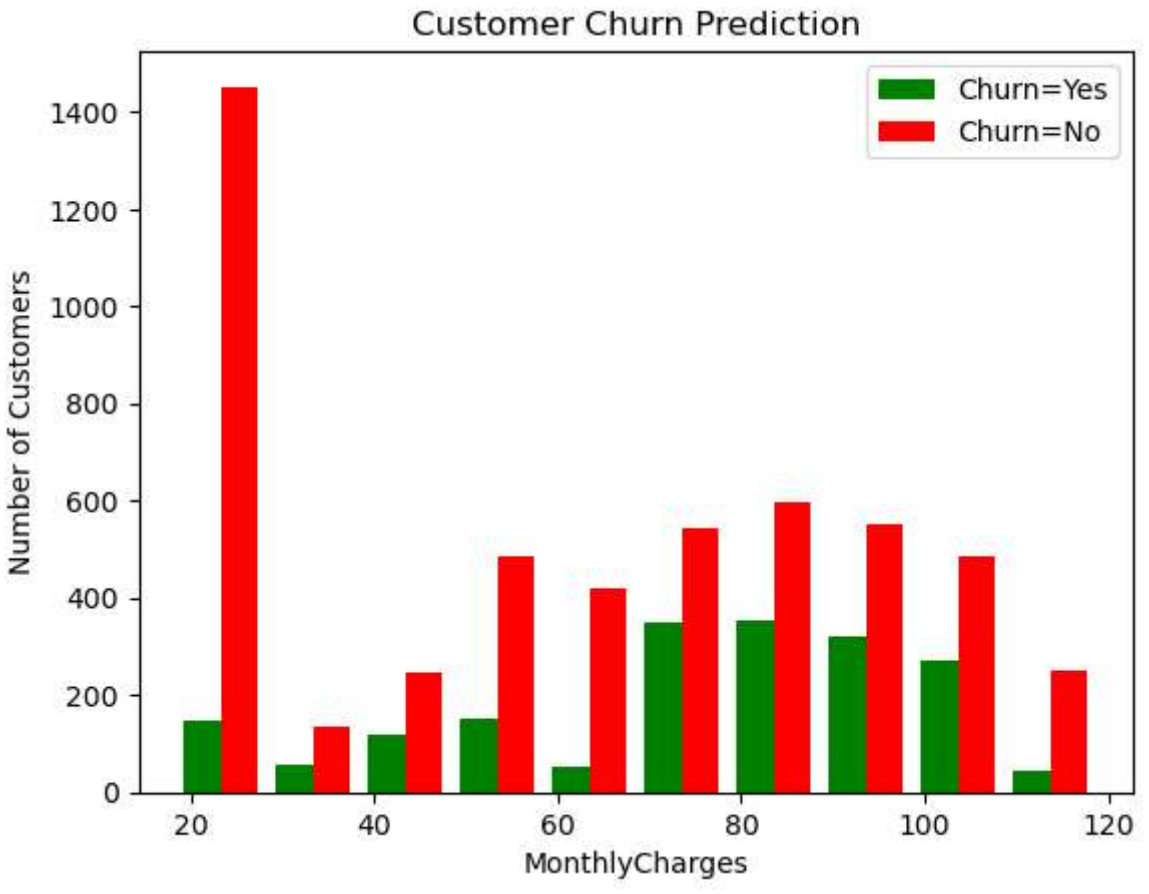
In [45]:
```
tenure_churn_no=df1[df1.Churn=='No'].tenure
tenure_churn_yes=df1[df1.Churn=='Yes'].tenure
plt.xlabel('tenure')
plt.ylabel('Number of Customers')
plt.title('Customer Churn Prediction')

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

Out[45]:
```
<matplotlib.legend.Legend at 0x1ad231918e0>
```



In [46]:
```
mc_churn_no=df1[df1.Churn=='No'].MonthlyCharges
mc_churn_yes=df1[df1.Churn=='Yes'].MonthlyCharges
plt.xlabel('MonthlyCharges')
plt.ylabel('Number of Customers')
plt.title('Customer Churn Prediction')

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

Out[46]: <matplotlib.legend.Legend at 0x1ad23184910>



In [51]:
```
def print_unique_col_values(df):
    for column in df:
        print(f'{column}:{df[column].unique()}')
```

In [52]:
```
print_unique_col_values(df1)
```

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

In [53]:

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

```
C:\Users\KIRAN\AppData\Local\Temp\ipykernel_17080\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#retu
rning-a-view-versus-a-copy
  df1.replace('No internet service','No',inplace=True)
C:\Users\KIRAN\AppData\Local\Temp\ipykernel_17080\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#retu
rning-a-view-versus-a-copy
  df1.replace('No phone service','No',inplace=True)
```

In [54]:
```
print_unique_col_values(df1)
```

```
gender:['Female' 'Male']
SeniorCitizen:[0 1]
Partner:['Yes' 'No']
Dependents:['No' 'Yes']
tenure:[ 1 34  2 45  8 22 10 28 62 13 16 58 49 25 69 52 71 21 12 30 47 72 17 27
  5 46 11 70 63 43 15 60 18 66  9  3 31 50 64 56  7 42 35 48 29 65 38 68
 32 55 37 36 41  6  4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26 39]
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)']
MonthlyCharges:[29.85 56.95 53.85 ... 63.1  44.2  78.7 ]
TotalCharges:[  29.85 1889.5   108.15 ...  346.45  306.6  6844.5 ]
Churn:['No' 'Yes']
```

In [59]:

```
yes_no_columns=['Partner','Dependents','PhoneService','MultipleLines','OnlineSecurity','OnlineBackup','DeviceProt
                'TechSupport','StreamingTV','StreamingMovies','PaperlessBilling','Churn']

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

```
C:\Users\KIRAN\AppData\Local\Temp\ipykernel_17080\2999861550.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#retu
rning-a-view-versus-a-copy
  df1[col].replace({'Yes':1, 'No':0},inplace=True)
```

In [60]:
```
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 [61]:
```
df1['gender'].replace({'Female':1,'Male':0},inplace=True)
```

```
C:\Users\KIRAN\AppData\Local\Temp\ipykernel_17080\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#retu
rning-a-view-versus-a-copy
  df1['gender'].replace({'Female':1,'Male':0},inplace=True)
```

In [62]:
```
for col in df1:
    print(f'{col}:{df1[col].unique()}')
```

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

In [63]:
```
pd.get_dummies(data=df1,columns=['InternetService'])
```

Out[63]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtecti
0	1	0	1	0	1	0	0	0	1	
1	0	0	0	0	34	1	0	1	0	
2	0	0	0	0	2	1	0	1	1	
3	0	0	0	0	45	0	0	1	0	
4	1	0	0	0	2	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	
7038	0	0	1	1	24	1	1	1	0	
7039	1	0	1	1	72	1	1	0	1	
7040	1	0	1	1	11	0	0	1	0	
7041	0	1	1	0	4	1	1	0	0	
7042	0	0	0	0	66	1	0	1	0	

7032 rows × 22 columns

In [64]:
```
df2=pd.get_dummies(data=df1,columns=['InternetService','Contract','PaymentMethod'])
df2.columns
```

Out[64]:
```
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')
```

In [65]:
```
df1.sample(4)
```

Out[65]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup
4067	1	0	1	0	68	1	0	DSL	1	
863	1	0	0	0	3	1	0	Fiber optic	0	
4772	1	0	1	1	69	1	0	DSL	0	
2733	1	0	1	0	1	1	0	DSL	0	

In [67]:
```
df2.dtypes
```

Out[67]:
```
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
```

In [68]:
```
df2.sample(4)
```

Out[68]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtecti
5099	1	0	0	1	11	1	0	0	0	
269	1	0	1	0	71	1	1	0	0	
6997	1	0	1	0	27	1	0	0	1	
2396	1	1	1	0	38	1	1	1	1	

4 rows × 27 columns

In [69]:
```
df2.dtypes
```

Out[69]:
```
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
```

In [73]:
```
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 [74]:
```
df2.sample(3)
```

Out[74]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtec
3415	1	0	1	1	0.943662	1	1	0	0	
4162	1	0	1	1	1.000000	0	0	1	1	
4175	0	0	1	1	0.845070	1	0	0	0	

3 rows × 27 columns

In [76]:

```
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]
```

In [84]:
```
X=df2.drop('Churn',axis='columns')
y=df2['Churn']
```

In [85]:
```
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)
```

In [86]:
```
X_train.shape
```

Out[86]:
```
(5625, 26)
```

In [88]:
```
X_test.shape
```

Out[88]:
```
(1407, 26)
```

In [89]:
```
len(X_train.columns)
```

Out[89]:
```
26
```

In [94]:
```
import tensorflow as tf
from tensorflow import keras

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

])

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

```
Epoch 1/100
176/176 [==============================] - 1s 2ms/step - loss: 0.5159 - accuracy: 0.7355
Epoch 2/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4328 - accuracy: 0.7972
Epoch 3/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4207 - accuracy: 0.7991
Epoch 4/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4166 - accuracy: 0.8005
Epoch 5/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4141 - accuracy: 0.8034
Epoch 6/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4126 - accuracy: 0.8059
Epoch 7/100
176/176 [==============================] - 1s 3ms/step - loss: 0.4113 - accuracy: 0.8064
Epoch 8/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4099 - accuracy: 0.8034
Epoch 9/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4084 - accuracy: 0.8085
Epoch 10/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4083 - accuracy: 0.8096
Epoch 11/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4074 - accuracy: 0.8094
Epoch 12/100
176/176 [==============================] - 0s 3ms/step - loss: 0.4061 - accuracy: 0.8096
Epoch 13/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4059 - accuracy: 0.8080
Epoch 14/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4044 - accuracy: 0.8114
Epoch 15/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4043 - accuracy: 0.8091
Epoch 16/100
176/176 [==============================] - 0s 3ms/step - loss: 0.4036 - accuracy: 0.8119
Epoch 17/100
176/176 [==============================] - 1s 4ms/step - loss: 0.4040 - accuracy: 0.8123
Epoch 18/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4025 - accuracy: 0.8108
Epoch 19/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4027 - accuracy: 0.8119
```

```
Epoch 20/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4015 - accuracy: 0.8126
Epoch 21/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4012 - accuracy: 0.8126
Epoch 22/100
176/176 [==============================] - 0s 2ms/step - loss: 0.4002 - accuracy: 0.8128
Epoch 23/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3996 - accuracy: 0.8103
Epoch 24/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3998 - accuracy: 0.8124
Epoch 25/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3994 - accuracy: 0.8116
Epoch 26/100
176/176 [==============================] - 0s 3ms/step - loss: 0.3989 - accuracy: 0.8149
Epoch 27/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3985 - accuracy: 0.8144
Epoch 28/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3984 - accuracy: 0.8128
Epoch 29/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3975 - accuracy: 0.8142
Epoch 30/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3973 - accuracy: 0.8142
Epoch 31/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3971 - accuracy: 0.8133
Epoch 32/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3963 - accuracy: 0.8137
Epoch 33/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3965 - accuracy: 0.8156
Epoch 34/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3957 - accuracy: 0.8171
Epoch 35/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3954 - accuracy: 0.8132
Epoch 36/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3951 - accuracy: 0.8148
Epoch 37/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3945 - accuracy: 0.8140
Epoch 38/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3935 - accuracy: 0.8146
```

```
Epoch 39/100
176/176 [==============================] - 0s 3ms/step - loss: 0.3935 - accuracy: 0.8172
Epoch 40/100
176/176 [==============================] - 0s 3ms/step - loss: 0.3936 - accuracy: 0.8139
Epoch 41/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3932 - accuracy: 0.8156
Epoch 42/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3930 - accuracy: 0.8156
Epoch 43/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3930 - accuracy: 0.8160
Epoch 44/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3925 - accuracy: 0.8149
Epoch 45/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3923 - accuracy: 0.8162
Epoch 46/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3918 - accuracy: 0.8169
Epoch 47/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3910 - accuracy: 0.8188
Epoch 48/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3907 - accuracy: 0.8151
Epoch 49/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3906 - accuracy: 0.8155
Epoch 50/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3902 - accuracy: 0.8164
Epoch 51/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3904 - accuracy: 0.8181
Epoch 52/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3896 - accuracy: 0.8176
Epoch 53/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3897 - accuracy: 0.8151
Epoch 54/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3898 - accuracy: 0.8172
Epoch 55/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3895 - accuracy: 0.8172
Epoch 56/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3883 - accuracy: 0.8187
Epoch 57/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3889 - accuracy: 0.8174
```

```
Epoch 58/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3883 - accuracy: 0.8176
Epoch 59/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3886 - accuracy: 0.8174
Epoch 60/100
176/176 [==============================] - 0s 3ms/step - loss: 0.3879 - accuracy: 0.8185
Epoch 61/100
176/176 [==============================] - 0s 3ms/step - loss: 0.3878 - accuracy: 0.8187
Epoch 62/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3872 - accuracy: 0.8185
Epoch 63/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3874 - accuracy: 0.8203
Epoch 64/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3867 - accuracy: 0.8162
Epoch 65/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3860 - accuracy: 0.8178
Epoch 66/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3871 - accuracy: 0.8178
Epoch 67/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3857 - accuracy: 0.8156
Epoch 68/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3861 - accuracy: 0.8181
Epoch 69/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3862 - accuracy: 0.8199
Epoch 70/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3850 - accuracy: 0.8181
Epoch 71/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3861 - accuracy: 0.8185
Epoch 72/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3853 - accuracy: 0.8180
Epoch 73/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3851 - accuracy: 0.8190
Epoch 74/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3846 - accuracy: 0.8178
Epoch 75/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3847 - accuracy: 0.8199
Epoch 76/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3843 - accuracy: 0.8183
```

```
Epoch 77/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3851 - accuracy: 0.8176
Epoch 78/100
176/176 [==============================] - 0s 3ms/step - loss: 0.3839 - accuracy: 0.8192
Epoch 79/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3841 - accuracy: 0.8183
Epoch 80/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3839 - accuracy: 0.8213
Epoch 81/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3837 - accuracy: 0.8185
Epoch 82/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3838 - accuracy: 0.8160
Epoch 83/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3830 - accuracy: 0.8204
Epoch 84/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3832 - accuracy: 0.8181
Epoch 85/100
176/176 [==============================] - 0s 3ms/step - loss: 0.3827 - accuracy: 0.8192
Epoch 86/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3832 - accuracy: 0.8196
Epoch 87/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3825 - accuracy: 0.8213
Epoch 88/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3828 - accuracy: 0.8183
Epoch 89/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3824 - accuracy: 0.8197
Epoch 90/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3825 - accuracy: 0.8165
Epoch 91/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3816 - accuracy: 0.8206
Epoch 92/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3817 - accuracy: 0.8201
Epoch 93/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3809 - accuracy: 0.8201
Epoch 94/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3807 - accuracy: 0.8208
Epoch 95/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3820 - accuracy: 0.8213
```

```
Epoch 96/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3815 - accuracy: 0.8194
Epoch 97/100
176/176 [==============================] - 1s 3ms/step - loss: 0.3812 - accuracy: 0.8208
Epoch 98/100
176/176 [==============================] - 0s 3ms/step - loss: 0.3814 - accuracy: 0.8190
Epoch 99/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3810 - accuracy: 0.8203
Epoch 100/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3812 - accuracy: 0.8194
```

Out[94]:
```
<keras.callbacks.History at 0x1ad33cda160>
```

In [95]:
```
yp=model.predict(X_test)
yp[:5]
```

```
44/44 [==============================] - 12s 3ms/step
```

Out[95]:
```
array([[0.17090558],
       [0.50734645],
       [0.01317747],
       [0.8144225 ],
       [0.4282514 ]], dtype=float32)
```

In [96]:
```
y_test[:5]
```

Out[96]:
```
2660    0
744     0
5579    1
64      1
3287    1
Name: Churn, dtype: int64
```

In [98]:
```
y_pred=[]
for element in yp:
    if element > 0.5:
        y_pred.append(1)
    else:
         y_pred.append(0)
```

In [99]:
```
y_pred[:10]
```

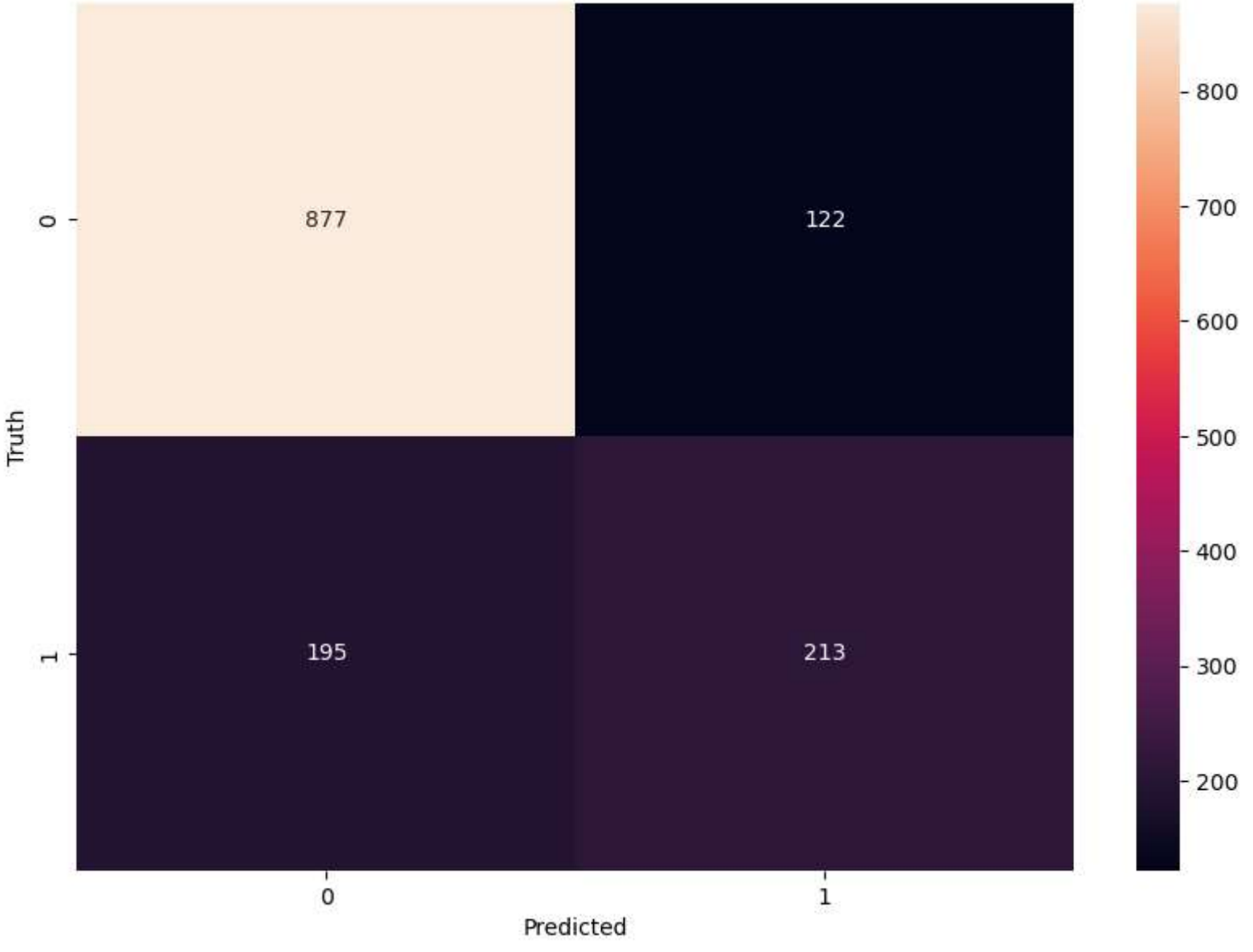
Out[99]:
```
[0, 1, 0, 1, 0, 1, 0, 0, 0, 0]
```

In [100…
```
from sklearn.metrics import confusion_matrix,classification_report

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

```
              precision    recall  f1-score   support

           0       0.82      0.88      0.85       999
           1       0.64      0.52      0.57       408

    accuracy                           0.77      1407
   macro avg       0.73      0.70      0.71      1407
weighted avg       0.77      0.77      0.77      1407
```

In [102…
```
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[102]:
```
Text(95.72222222222221, 0.5, 'Truth')
```


877
122
195
213
0
1
Truth
0
1
Predicted
800
700
600
500
400
300
200

Accuracy

In [119…
```
round((877+213)/(877+122+195+213),2)*100
```

Out[119]:
```
77.0
```

Precision for 0,1 class

In [120…
```
round(877/(877+195),2)
```

Out[120]:
```
0.82
```

In [115…
```
round(213/(213+122),2)
```

Out[115]:
```
0.64
```

Recall for 0,1 class

In [116…
```
round(877/(877+122),2)
```

Out[116]:
```
0.88
```

In [118…
```
round(213/(213+195),2)
```

Out[118]:
```
0.52
```

Exercise

Take this dataset fro bank customer churn prediction : https://www.kaggle.com/barelydedicated/bank-customer-churn-modeling

1. Build a deep learning model to predict churn rate at bank
2. Once model is built, print classification report and analyze precision, recall and f1-score

In [ ]: