Customer Churn Prediction

In [1]:

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

In [2]:

```
df = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")
df.sample(5)
```

Out[2]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLi
5521	3863-QSTYI	Male	0	No	No	59	Yes	
480	0486-LGCCH	Male	0	Yes	Yes	11	Yes	
3929	9702-AIUJO	Male	0	Yes	Yes	50	Yes	
810	2239-JALAW	Male	0	No	No	58	Yes	
1822	4911-BANWH	Female	0	No	Yes	31	Yes	

5 rows × 21 columns

In [3]:

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

In [4]:

```
df.dtypes
```

Out[4]:

```
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 [5]:

```
df.TotalCharges
```

Out[5]:

```
0         29.85
1        1889.5
2        108.15
3       1840.75
4        151.65
         ...
7038     1990.5
7039     7362.9
7040     346.45
7041      306.6
7042     6844.5
Name: TotalCharges, Length: 7043, dtype: object
```

In [6]:

```
df.TotalCharges.values
```

Out[6]:

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

In [7]:

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

Out[7]:

```
11
```

In [8]:

```
df.iloc[488].TotalCharges
```

Out[8]:

```
' '
```

In [9]:

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

Out[9]:

```
(7032, 20)
```

In [55]:

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

Out[55]:

```
(7032, 20)
```

In [56]:

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

```
C:\Users\Lenovo\anaconda3\lib\site-packages\pandas\core\generic.py:5516: Set
tingWithCopyWarning:
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 (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  self[name] = value
```

In [33]:

```
df1.dtypes
```

Out[33]:

```
gender              object
SeniorCitizen        int64
Partner             object
Dependents          object
tenure               int64
PhoneService        object
MultipleLines       object
InternetService     object
OnlineSecurity      object
OnlineBackup        object
DeviceProtection    object
TechSupport         object
StreamingTV         object
StreamingMovies     object
Contract            object
PaperlessBilling    object
PaymentMethod       object
MonthlyCharges     float64
TotalCharges       float64
Churn               object
dtype: object
```

In [57]:

```
df1.TotalCharges.values
```

Out[57]:

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

In [58]:

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

Out[58]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Internet
0	Female	0	Yes	No	1	No	No phone service	
1	Male	0	No	No	34	Yes	No	
3	Male	0	No	No	45	No	No phone service	
6	Male	0	No	Yes	22	Yes	Yes	Fib
7	Female	0	No	No	10	No	No phone service	
...	...	...	...	...	...	...	...	
7037	Female	0	No	No	72	Yes	No	
7038	Male	0	Yes	Yes	24	Yes	Yes	
7039	Female	0	Yes	Yes	72	Yes	Yes	Fib
7040	Female	0	Yes	Yes	11	No	No phone service	
7042	Male	0	No	No	66	Yes	No	Fib

5163 rows × 20 columns

In [59]:

```
df1[df1.Churn=='Yes']
```

Out[59]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Internet
2	Male	0	No	No	2	Yes	No	
4	Female	0	No	No	2	Yes	No	Fib
5	Female	0	No	No	8	Yes	Yes	Fib
8	Female	0	Yes	No	28	Yes	Yes	Fib
13	Male	0	No	No	49	Yes	Yes	Fib
...	...	...	...	...	...	...	...	
7021	Male	0	No	No	12	Yes	No	
7026	Female	0	No	No	9	Yes	No	
7032	Male	1	No	No	1	Yes	Yes	Fib
7034	Female	0	No	No	67	Yes	Yes	Fib
7041	Male	1	Yes	No	4	Yes	Yes	Fib

1869 rows × 20 columns

In [60]:

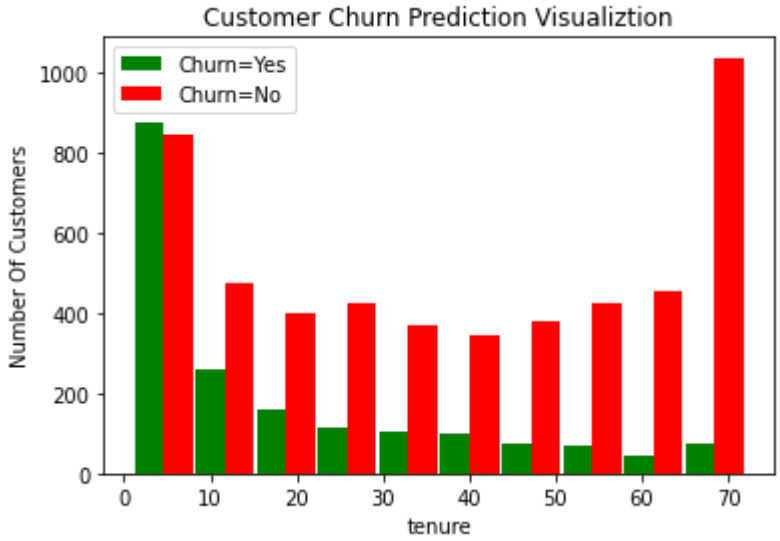
```
tenure_churn_no = df1[df1.Churn=='No'].tenure
tenure_churn_yes = df1[df1.Churn=='Yes'].tenure

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


plt.hist([tenure_churn_yes, tenure_churn_no], rwidth=0.95, color=['green','red'],label=['Ch
plt.legend()
```

Out[60]:

```
<matplotlib.legend.Legend at 0x274dee11160>
```



In [61]:

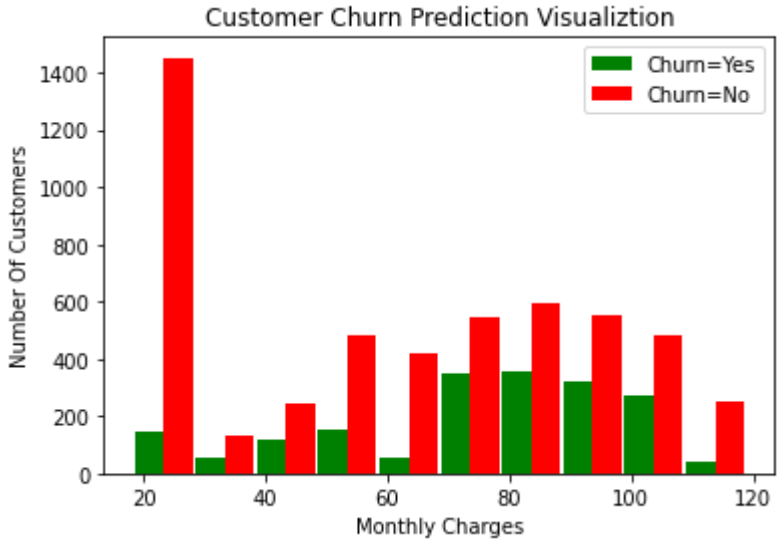
```
mc_churn_no = df1[df1.Churn=='No'].MonthlyCharges
mc_churn_yes = df1[df1.Churn=='Yes'].MonthlyCharges

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


plt.hist([mc_churn_yes, mc_churn_no], rwidth=0.95, color=['green','red'],label=['Churn=Yes'
plt.legend()
```

Out[61]:

```
<matplotlib.legend.Legend at 0x274c5aae130>
```



In [62]:

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

In [63]:

```
print_unique_col_values(df)
```

```
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 (automati
c)'
 'Credit card (automatic)']
TotalCharges: ['29.85' '1889.5' '108.15' ... '346.45' '306.6' '6844.5']
Churn: ['No' 'Yes']
```

In [64]:

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

```
C:\Users\Lenovo\anaconda3\lib\site-packages\pandas\core\frame.py:5238: Setti
ngWithCopyWarning:
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 (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  return super().replace(
```

In [65]:

```
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 (automati
c)'
 'Credit card (automatic)']
Churn: ['No' 'Yes']
```

In [66]:

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

```
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 1
7 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 (automati
c)'
 '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 [67]:

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

Out[67]:

```
array(['Female', 'Male'], dtype=object)
```

In [176]:

```
df1.Churn.unique()
```

Out[176]:

```
array(['No', 'Yes'], dtype=object)
```

In [177]:

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

```
C:\Users\Lenovo\anaconda3\lib\site-packages\pandas\core\generic.py:6619: Set
tingWithCopyWarning:
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 (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  return self._update_inplace(result)
```

In [178]:

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

In [179]:

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

In [180]:

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

In [181]:

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

In [182]:

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

In [183]:

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

In [184]:

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

In [185]:

```
df1['DeviceProtection'].replace({'Yes':1,'No':0},inplace=True)
df1['Churn'].replace({'Yes':1,'No':0},inplace=True)
```

In [186]:

```
df1['TechSupport'].replace({'Yes':1,'No':0},inplace=True)
df1['StreamingTV'].replace({'Yes':1,'No':0},inplace=True)
df1['StreamingMovies'].replace({'Yes':1,'No':0},inplace=True)
df1['PaperlessBilling'].replace({'Yes':1,'No':0},inplace=True)
```

In [187]:

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

Out[187]:

```
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 [188]:

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

Out[188]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineS
863	1	0	0	0	3	1	0	
4104	0	0	1	0	8	1	0	
6355	1	0	0	0	68	1	1	
3678	0	0	1	1	16	1	1	
1456	1	0	1	0	16	1	1	

5 rows × 27 columns

In [189]:

```
df2.dtypes
```

Out[189]:

```
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 [190]:

```
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 [191]:

```
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 [192]:

```
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)
```

In [193]:

```
X_train.shape
```

Out[193]:

```
(5625, 26)
```

In [194]:

```
X_test.shape
```

Out[194]:

```
(1407, 26)
```

In [195]:

```
X_train = X_train.astype(float)
```

In [ ]:

In [ ]:

In [ ]:

In [196]:

```
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')
])



model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.fit(X_train, y_train, epochs=100)
```

```
176/176 [==============================] - 0s 2ms/step - loss: 0.3608 - a
ccuracy: 0.8292
Epoch 85/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3592 - a
ccuracy: 0.8304
Epoch 86/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3582 - a
ccuracy: 0.8299
Epoch 87/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3583 - a
ccuracy: 0.8357
Epoch 88/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3575 - a
ccuracy: 0.8334
Epoch 89/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3570 - a
ccuracy: 0.8320
Epoch 90/100
176/176 [==============================] - 0s 2ms/step - loss: 0.3548 - a
ccuracy: 0.8345
```

In [197]:

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

```
44/44 [==============================] - 0s 1ms/step - loss: 0.4905 - accura
cy: 0.7726
```

Out[197]:

```
[0.49049249291419983, 0.7725657224655151]
```

In [200]:

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

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

Out[200]:

```
array([[0.28224123],
       [0.4679681 ],
       [0.00627228],
       [0.8261105 ],
       [0.5812232 ]], dtype=float32)
```

In [201]:

```
y_pred = []
for element in yp:
    if element > 0.5:
        y_pred.append(1)
    else:
        y_pred.append(0)
y_pred[:10]
[0, 0, 0, 1, 0, 1, 0, 0, 0, 0]
y_test[:10]
```

Out[201]:

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

In [202]:

```
from sklearn.metrics import confusion_matrix , classification_report

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

```
              precision    recall  f1-score   support

           0       0.81      0.88      0.85       999
           1       0.64      0.50      0.56       408

    accuracy                           0.77      1407
   macro avg       0.73      0.69      0.70      1407
weighted avg       0.76      0.77      0.76      1407
```

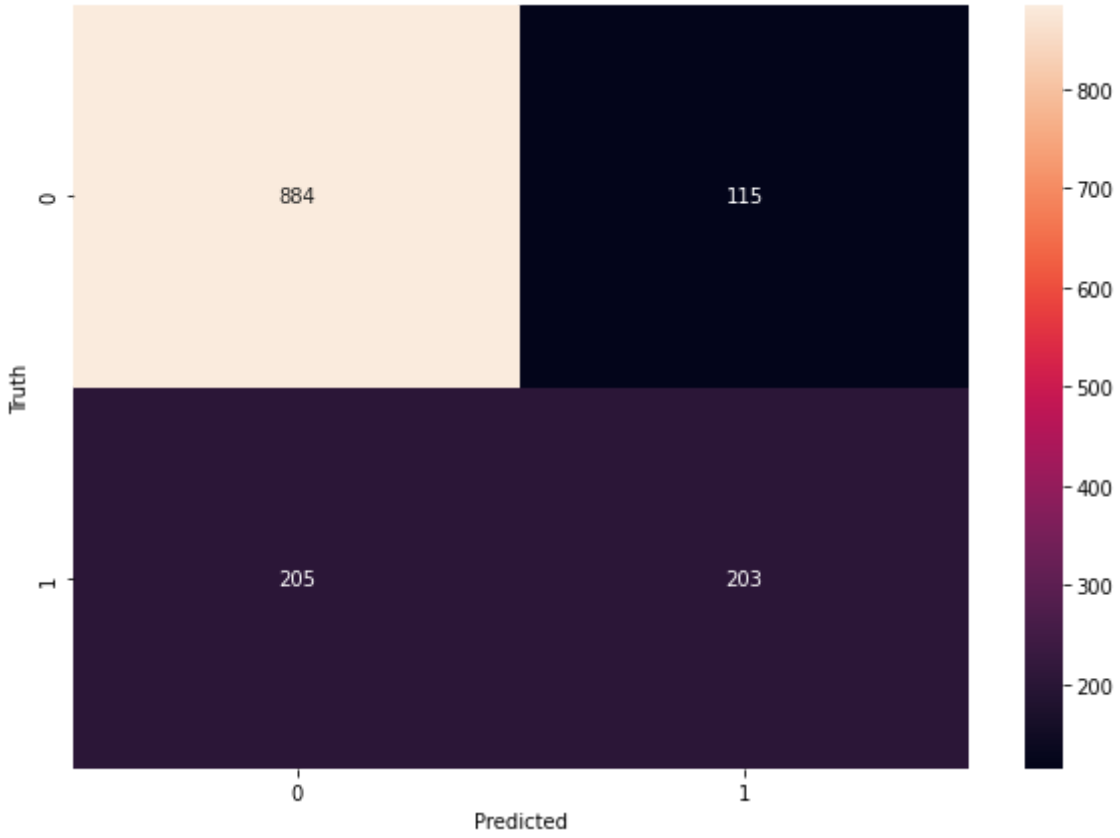
In [203]:

```
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')
```

Out[203]:

```
Text(69.0, 0.5, 'Truth')
```



In [ ]: