In [1]:

```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings("ignore")
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [4]:

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

Out[4]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service
1	5575-GNVDE	Male	0	No	No	34	Yes	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service
4	9237-HQITU	Female	0	No	No	2	Yes	No

5 rows × 21 columns

In [5]:

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

In [6]:

```
df.dtypes
```

Out[6]:

```
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
```

OBSERVATION - 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

Now using to_numeric method I will change the object into numerical data

In [7]:

```
pd.to_numeric(df.TotalCharges)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
pandas\_libs\lib.pyx in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string " "

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
<ipython-input-7-06ba430a4ba5> in <module>
----> 1 pd.to_numeric(df.TotalCharges)

~\anaconda3\lib\site-packages\pandas\core\tools\numeric.py in to_numeric(ar
g, errors, downcast)
    147         coerce_numeric = errors not in ("ignore", "raise")
    148         try:
--> 149             values = lib.maybe_convert_numeric(
    150                 values, set(), coerce_numeric=coerce_numeric
    151             )

pandas\_libs\lib.pyx in pandas._libs.lib.maybe_convert_numeric()

ValueError: Unable to parse string " " at position 488
```

OBSERVATION - Error is shown because some values seems to be not numbers but blank string in the TotalCharge column which pandas is not able to convert to numerical values.

In [8]:

```
# THIS WILL CONVERT OBJECT WITHOUT BLANK STRING TO FLOAT WHILE BLANK STRING WILL BE CONVERT
pd.to_numeric(df.TotalCharges,errors='coerce')
```

Out[8]:

```
0         29.85
1       1889.50
2        108.15
3       1840.75
4        151.65
         ...
7038    1990.50
7039    7362.90
7040     346.45
7041     306.60
7042    6844.50
Name: TotalCharges, Length: 7043, dtype: float64
```

LET'S SAY THE I WANT TO SEE THE INDEXES OF ROWS WITH NULL VALUES

In [9]:

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

Out[9]:

```
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 [10]:

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

Out[10]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Internet
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 [11]:

```
# iloc - integer location
df.iloc[488]
```

Out[11]:

```
gender                                   Female
SeniorCitizen                                 0
Partner                                     Yes
Dependents                                  Yes
tenure                                        0
PhoneService                                 No
MultipleLines                 No phone service
InternetService                              DSL
OnlineSecurity                              Yes
OnlineBackup                                 No
DeviceProtection                            Yes
TechSupport                                 Yes
StreamingTV                                 Yes
StreamingMovies                              No
Contract                                Two year
PaperlessBilling                            Yes
PaymentMethod       Bank transfer (automatic)
MonthlyCharges                              52.55
TotalCharges
Churn                                        No
Name: 488, dtype: object
```

In [12]:

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

Out[12]:

```
' '
```

In [13]:

```
print(f"Total rows: {df.shape[0]}")
print(f"Rows with null values: {df[df.TotalCharges==' '].shape[0]}")
```

```
Total rows: 7043
Rows with null values: 11
```

In [14]:

```
# NOW LETS DROP BLANK SPACES AS ONLY 11 ROWS HAVE NULL VALUES
df1 = df[df.TotalCharges!=' ']
print(df1.shape)
df1.head()
```

```
(7032, 20)
```

Out[14]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetSer
0	Female	0	Yes	No	1	No	No phone service	
1	Male	0	No	No	34	Yes	No	
2	Male	0	No	No	2	Yes	No	
3	Male	0	No	No	45	No	No phone service	
4	Female	0	No	No	2	Yes	No	Fiber c

In [15]:

```
# OBJECT -> NUMERIC
df1.TotalCharges = pd.to_numeric(df1.TotalCharges)
```

In [16]:

```
df1.dtypes
```

Out[16]:

```
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 [17]:

```
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 (automati
c)'
 '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

In [30]:

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

In [31]:

```
print_unique_col_values(df1)
```

```
gender: ['Female' 'Male']
InternetService: ['DSL' 'Fiber optic' 'No']
Contract: ['Month-to-month' 'One year' 'Two year']
PaymentMethod: ['Electronic check' 'Mailed check' 'Bank transfer (automati
c)'
 'Credit card (automatic)']
```

In [32]:

```
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 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: [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 (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: [0 1]
```

One hot encoding for categorical columns

In [33]:

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

Out[33]:

```
Index(['SeniorCitizen', 'Partner', 'Dependents', 'tenure', 'PhoneService',
       'MultipleLines', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtectio
n',
       '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',
       'gender_Female', 'gender_Male'],
      dtype='object')
```

In [34]:

```
df2.head(5)
```

Out[34]:

	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	Onl
0	0	1	0	1	0	0	0	
1	0	0	0	34	1	0	1	
2	0	0	0	2	1	0	1	
3	0	0	0	45	0	0	1	
4	0	0	0	2	1	0	0	

5 rows × 28 columns

In [35]:

```
df2.dtypes
```

Out[35]:

```
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
gender_Female                              uint8
gender_Male                                uint8
dtype: object
```

In [36]:

```
df.describe()
```

Out[36]:

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

In [37]:

```
df.describe().columns
```

Out[37]:

```
Index(['SeniorCitizen', 'tenure', 'MonthlyCharges'], dtype='object')
```

In [38]:

```
cols_to_scale = []
for i in df.describe().columns:
    cols_to_scale.append(i)
cols_to_scale
```

Out[38]:

```
['SeniorCitizen', 'tenure', 'MonthlyCharges']
```

In [39]:

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

In [40]:

```
df2.describe()
```

Out[40]:

	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Onlir
count	7032.000000	7032.000000	7032.000000	7032.000000	7032.000000	7032.000000	70
mean	0.162400	0.482509	0.298493	0.442560	0.903299	0.421928	
std	0.368844	0.499729	0.457629	0.345708	0.295571	0.493902	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.112676	1.000000	0.000000	
50%	0.000000	0.000000	0.000000	0.394366	1.000000	0.000000	
75%	0.000000	1.000000	1.000000	0.760563	1.000000	1.000000	
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	

8 rows × 28 columns

In [41]:

```
for col in df2:
    print(f'{col}: {df2[col].unique()}')
```

```
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: [  29.85 1889.5   108.15 ...  346.45  306.6  6844.5 ]
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]
gender_Female: [1 0]
gender_Male: [0 1]
```

Train test split

In [42]:

```
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 [43]:

```
X_train.shape, X_test.shape
```

Out[43]:

```
((5625, 27), (1407, 27))
```

BUIDLING A MODEL

In [44]:

```
import tensorflow as tf
from tensorflow import keras



model = keras.Sequential([
    keras.layers.Dense(26, input_shape=(27,), 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)
```

```
accuracy: 0.7783
Epoch 77/100
176/176 [==============================] - 0s 946us/step - loss: 0.5201 -
accuracy: 0.7890
Epoch 78/100
176/176 [==============================] - 0s 1ms/step - loss: 0.5867 - ac
curacy: 0.7829
Epoch 79/100
176/176 [==============================] - 0s 958us/step - loss: 0.5122 -
accuracy: 0.7904
Epoch 80/100
```

In [45]:

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

```
44/44 [==============================] - 0s 970us/step - loss: 0.5541 - accu
racy: 0.7832
```

Out[45]:

```
[0.5540950298309326, 0.783226728439331]
```

In [46]:

```
yp = model.predict(X_test)
```

In [47]:

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

In [49]:

```
from sklearn.metrics import confusion_matrix , classification_report

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

```
              precision    recall  f1-score   support

           0       0.80      0.92      0.86       999
           1       0.69      0.45      0.55       408

    accuracy                           0.78      1407
   macro avg       0.75      0.68      0.70      1407
weighted avg       0.77      0.78      0.77      1407
```

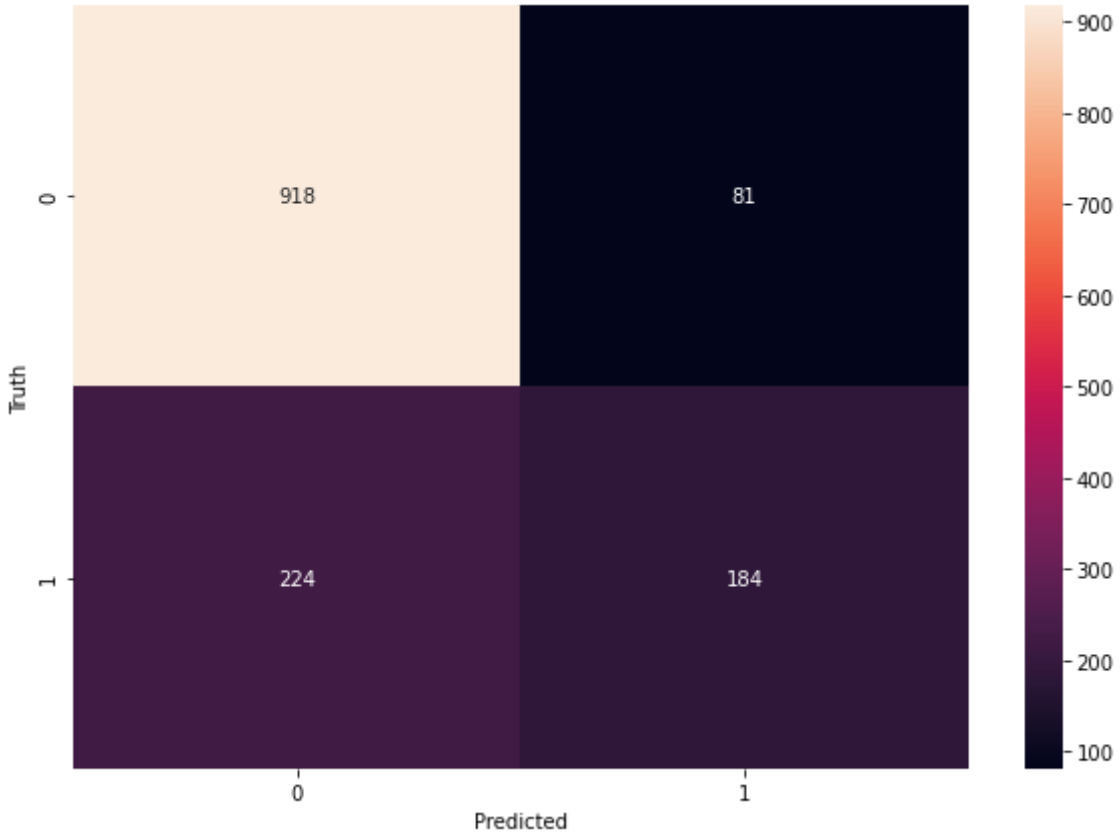
In [50]:

```
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[50]:

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



In [ ]: