Imarticus Learning

JUNE 2023

Customer Churn Prediction

Prepared By:

Dinesh Babu

Imarticus Learning, Coimbatore | +91-6374249560 | https://github.com/Dineshbabu1001

In [ ]:

```
!pip install pandas
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) h
ttps://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.p
kg.dev/colab-wheels/public/simple/)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-p
ackages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/p
ython3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/
dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.1
0/dist-packages (from pandas) (1.22.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist
-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

In [ ]:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as snp
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
```

In [ ]:

```
Churn_Data=pd.read_csv('Telco-Customer-Churn.csv')
```

In [ ]:

```
Churn_Data.sample(10)
```

Out[4]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	Multipl
6663	0674-EYYZV	Female	0	No	No	1	Yes	
2361	5103-MHMHY	Female	0	No	Yes	1	Yes	
6221	0042-JVWOJ	Male	0	No	No	26	Yes	
5150	7017-VFULY	Female	0	Yes	No	2	Yes	
6432	8221-EQDGL	Male	0	Yes	No	35	Yes	
4708	5181-OABFK	Female	0	Yes	Yes	56	Yes	
3099	5505-OVWQW	Female	0	No	No	17	Yes	
374	6862-CQUMB	Male	0	No	No	37	Yes	
3329	5366-OBVMR	Female	0	Yes	No	18	Yes	
6086	0916-KNFAJ	Male	0	Yes	No	61	Yes	

10 rows × 21 columns

In [ ]:

```
Churn_Data.drop('customerID',axis=1,inplace=True)
```

In [ ]:

```
Churn_Data.sample(8)
```

Out[6]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	Inter
6703	Female	0	No	No	2	No	No phone service	
6483	Male	0	Yes	Yes	13	Yes	No	
621	Female	0	No	No	62	Yes	Yes	
1023	Female	1	Yes	No	45	Yes	No	
4269	Male	0	Yes	No	17	Yes	Yes	
6668	Female	0	No	No	38	Yes	Yes	
5453	Male	0	Yes	No	60	Yes	No	
2688	Male	0	Yes	No	5	Yes	Yes	

In [ ]:

```
Churn_Data.dtypes
```

Out[7]:

```
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 [ ]:

```
#Convert 'Total Charges' column from object to numerical
Churn_Data['Totalcharges']=pd.to_numeric(Churn_Data['TotalCharges'],errors='coerce')
print(Churn_Data.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
Totalcharges       float64
dtype: object
```

In [ ]:

```
Churn_Data.shape
```

Out[9]:

```
(7043, 21)
```

In [ ]:

```
Churn_Data.drop_duplicates()
Churn_Data.shape
```

Out[10]:

```
(7043, 21)
```

In [ ]:

```
Churn_Data.isnull().sum()
```

Out[11]:

```
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
Totalcharges       11
dtype: int64
```

In [ ]:

```
# Replace empty strings or whitespace with NaN
Churn_Data['TotalCharges'].replace('', np.nan, inplace=True)
Churn_Data['TotalCharges'].replace(' ', np.nan, inplace=True)

# Drop rows with null values in the 'TotalCharges' column
Churn_Data.dropna(subset=['TotalCharges'], inplace=True)

print(Churn_Data.isnull().sum())
```

```
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
Totalcharges        0
dtype: int64
```

In [ ]:

```
Churn_Data.dropna(inplace=True)

print(Churn_Data.isnull().sum())
```

```
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
Totalcharges        0
dtype: int64
```

In [ ]:

```
Churn_Data.shape
```

Out[14]:

```
(7032, 21)
```

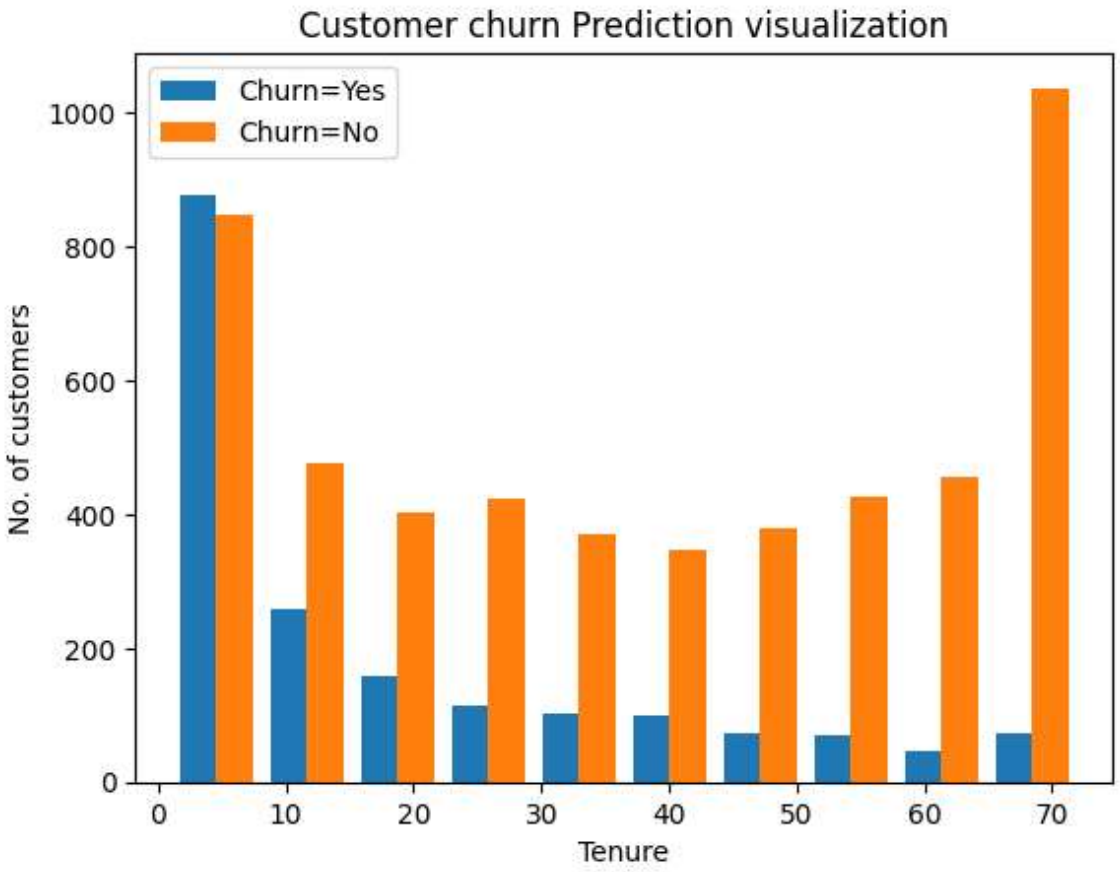
#EDA

In [ ]:

```
tenure_churn_no=Churn_Data[Churn_Data.Churn=='No'].tenure
tenure_churn_yes=Churn_Data[Churn_Data.Churn=='Yes'].tenure
plt.hist([tenure_churn_yes,tenure_churn_no],label=['Churn=Yes','Churn=No'])
plt.legend()
plt.xlabel('Tenure')
plt.ylabel('No. of customers')
plt.title ('Customer churn Prediction visualization')
```

Out[15]:

```
Text(0.5, 1.0, 'Customer churn Prediction visualization')
```

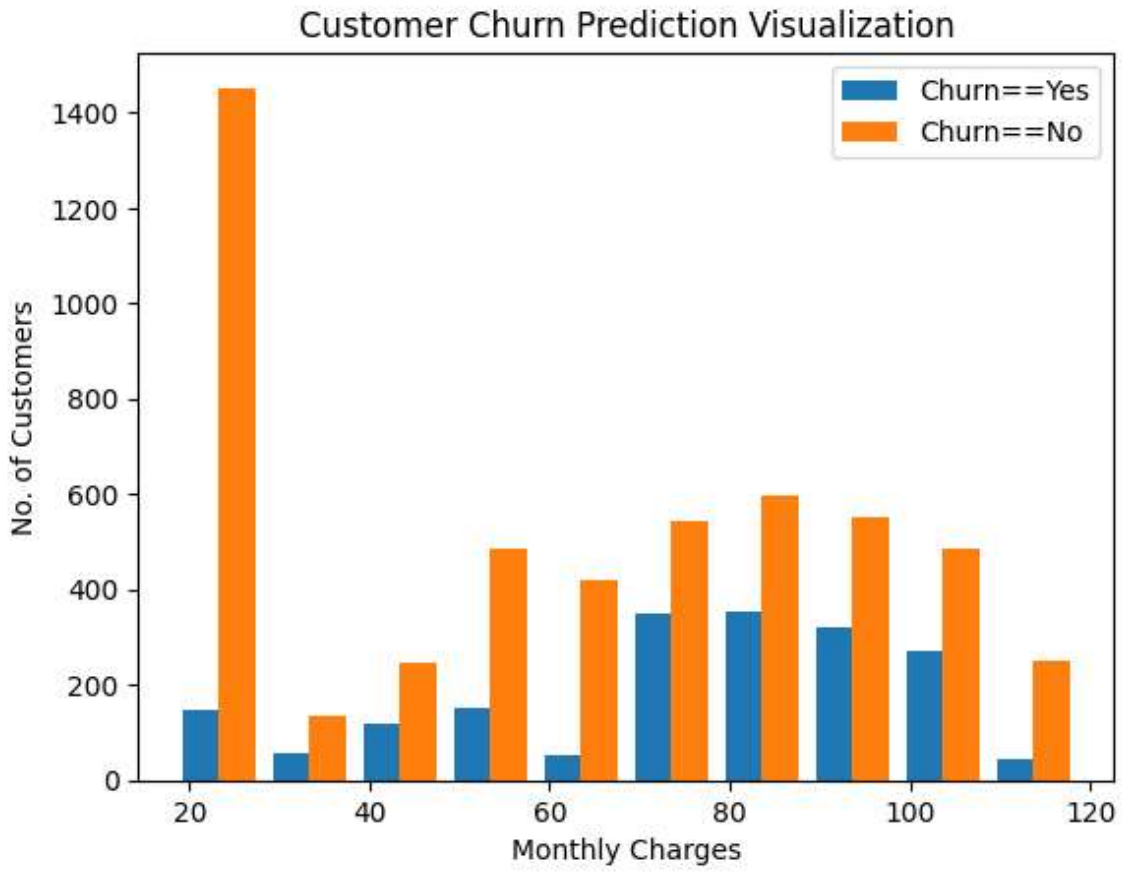


In [ ]:

```
monthly_charges_no=Churn_Data[Churn_Data.Churn=='No'].MonthlyCharges
monthly_charges_yes=Churn_Data[Churn_Data.Churn=='Yes'].MonthlyCharges
plt.hist([monthly_charges_yes,monthly_charges_no],
         label=['Churn==Yes','Churn==No'])
plt.legend()
plt.xlabel('Monthly Charges')
plt.ylabel('No. of Customers')
plt.title('Customer Churn Prediction Visualization')
```

Out[16]:

```
Text(0.5, 1.0, 'Customer Churn Prediction Visualization')
```

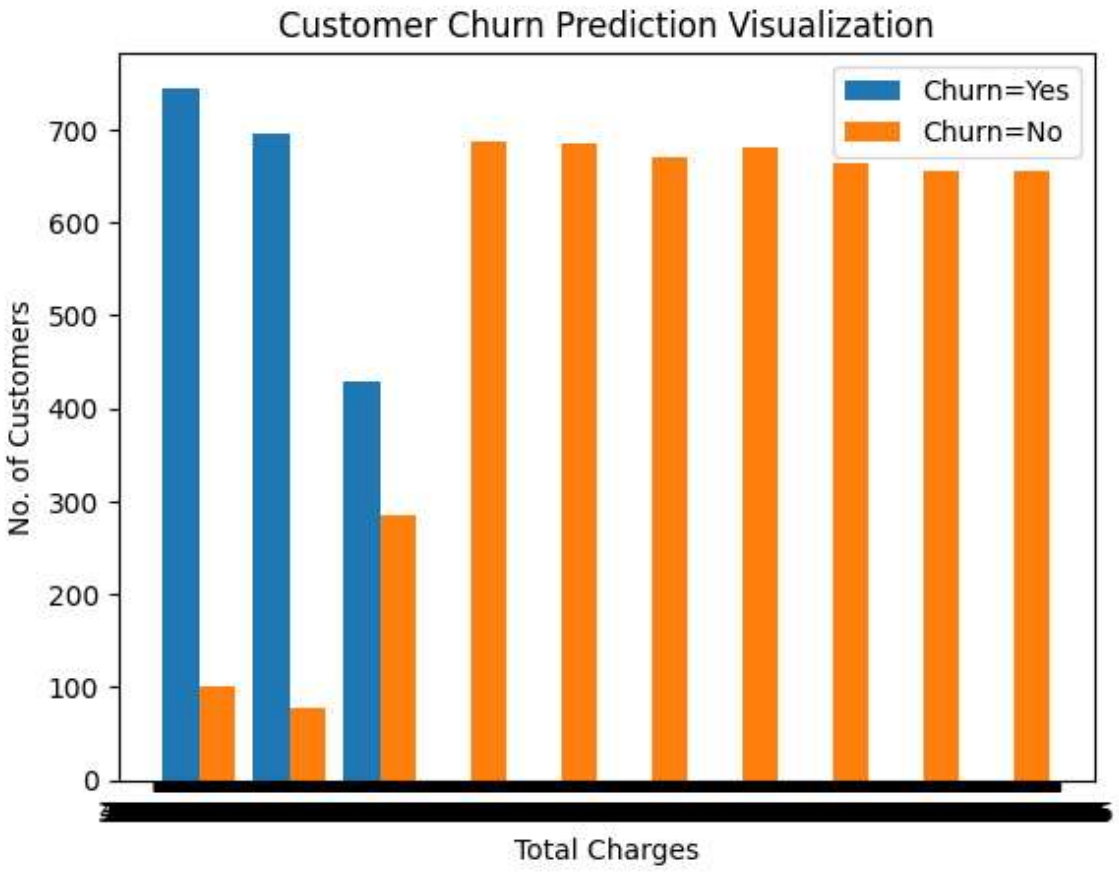


In [ ]:

```
Total_Charges_no=Churn_Data[Churn_Data.Churn=='No'].TotalCharges
Total_Charges_yes=Churn_Data[Churn_Data.Churn=='Yes'].TotalCharges
plt.hist([Total_Charges_yes,Total_Charges_no],label=['Churn=Yes','Churn=No'])
plt.legend()
plt.xlabel('Total Charges')
plt.ylabel('No. of Customers')
plt.title('Customer Churn Prediction Visualization')
```

Out[17]:

```
Text(0.5, 1.0, 'Customer Churn Prediction Visualization')
```



In [ ]:

```
Churn_Data.nunique()
```

Out[18]:

```
gender                2
SeniorCitizen         2
Partner               2
Dependents            2
tenure               72
PhoneService          2
MultipleLines         3
InternetService       3
OnlineSecurity        3
OnlineBackup          3
DeviceProtection      3
TechSupport           3
StreamingTV           3
StreamingMovies       3
Contract              3
PaperlessBilling      2
PaymentMethod         4
MonthlyCharges     1584
TotalCharges       6530
Churn                 2
Totalcharges       6530
dtype: int64
```

In [ ]:

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

Out[19]:

```
Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurit
y',
       'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
       'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMetho
d',
       'MonthlyCharges', 'TotalCharges', 'Churn', 'Totalcharges'],
      dtype='object')
```

In [ ]:

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

In [ ]:

```
Churn_Data['gender'].unique()
```

Out[21]:

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

In [ ]:

```
Churn_Data['Dependents'].unique()
```

Out[22]:

```
array([0, 1])
```

In [ ]:

```
Churn_Data_1=pd.get_dummies(data=Churn_Data,
                            columns=['InternetService','Contract','PaymentMethod'])
Churn_Data_1.columns
```

Out[23]:

```
Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
       'PhoneService', 'MultipleLines', 'OnlineSecurity', 'OnlineBackup',
       'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovie
s',
       'PaperlessBilling', 'MonthlyCharges', 'TotalCharges', 'Churn',
       'Totalcharges', 'InternetService_DSL', 'InternetService_Fiber opti
c',
       'InternetService_No', 'Contract_Month-to-month', 'Contract_One yea
r',
       'Contract_Two year', 'PaymentMethod_Bank transfer (automatic)',
       'PaymentMethod_Credit card (automatic)',
       'PaymentMethod_Electronic check', 'PaymentMethod_Mailed check'],
      dtype='object')
```

In [ ]:

```
Churn_Data_1.dtypes
```

Out[24]:

```
gender                                    object
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                              object
Churn                                      int64
Totalcharges                             float64
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 [ ]:

```
col_to_scale=['tenure','MonthlyCharges','TotalCharges']
scaler=MinMaxScaler()
Churn_Data_1[col_to_scale]=scaler.fit_transform(Churn_Data_1[col_to_scale])
```

In [ ]:

```
Churn_Data_1[col_to_scale]
```

Out[26]:

	tenure	MonthlyCharges	TotalCharges
0	0.000000	0.115423	0.001275
1	0.464789	0.385075	0.215867
2	0.014085	0.354229	0.010310
3	0.619718	0.239303	0.210241
4	0.014085	0.521891	0.015330
...	...	...	...
7038	0.323944	0.662189	0.227521
7039	1.000000	0.845274	0.847461
7040	0.140845	0.112935	0.037809
7041	0.042254	0.558706	0.033210
7042	0.915493	0.869652	0.787641

7032 rows × 3 columns

In [ ]:

```
Churn_Data_1= pd.get_dummies(Churn_Data_1, columns=['gender'])
Churn_Data_1
```

Out[41]:

	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecu
0	0	1	0	0.000000	0	0	
1	0	0	0	0.464789	1	0	
2	0	0	0	0.014085	1	0	
3	0	0	0	0.619718	0	0	
4	0	0	0	0.014085	1	0	
...	...	...	...	...	...	...	
7038	0	1	1	0.323944	1	1	
7039	0	1	1	1.000000	1	1	
7040	0	1	1	0.140845	0	0	
7041	1	1	0	0.042254	1	1	
7042	0	0	0	0.915493	1	0	

7032 rows × 29 columns

In [ ]:

```
X=Churn_Data_1.drop('Churn',
                    axis='columns')
y=Churn_Data_1['Churn']
```

In [ ]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=42)
```

In [ ]:

```
X_train.shape
```

Out[44]:

```
(4922, 28)
```

In [ ]:

```
X_test.shape
```

Out[45]:

```
(2110, 28)
```

In [ ]:

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

Out[46]:

```
28
```

In [ ]:

```
# Feature scaling using Min-Max Scaler
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [ ]:

```
model=keras.Sequential([
    keras.layers.Dense(20,input_shape=(28,),activation='relu'),
    keras.layers.Dense(1,activation='sigmoid')
])
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

In [ ]:

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

```
Epoch 1/100
154/154 [==============================] - 1s 1ms/step - loss: 5.1854
- accuracy: 0.6735
Epoch 2/100
154/154 [==============================] - 0s 1ms/step - loss: 0.6965
- accuracy: 0.7462
Epoch 3/100
154/154 [==============================] - 0s 1ms/step - loss: 0.6616
- accuracy: 0.7690
Epoch 4/100
154/154 [==============================] - 0s 1ms/step - loss: 0.5778
- accuracy: 0.7761
Epoch 5/100
154/154 [==============================] - 0s 1ms/step - loss: 0.6767
- accuracy: 0.7706
Epoch 6/100
154/154 [==============================] - 0s 1ms/step - loss: 0.7045
- accuracy: 0.7714
Epoch 7/100
```

In [ ]:

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

```
66/66 [==============================] - 0s 949us/step - loss: 0.6900 - a
ccuracy: 0.6986
```

Out[52]:

```
[0.6900379061698914, 0.6985781788825989]
```

In [ ]:

```
y_pred=model.predict(X_test)
y_pred[:2]
```

```
66/66 [==============================] - 0s 749us/step
```

Out[53]:

```
array([[0.01115342],
       [0.22705288]], dtype=float32)
```

In [ ]:

```
y_test
```

Out[54]:

```
2481    0
6784    0
6125    1
3052    0
4099    0
       ..
2763    0
6747    0
1700    0
1099    0
4720    0
Name: Churn, Length: 2110, dtype: int64
```

In [ ]:

```
y_prediction=[]
for elements in y_pred:
  if elements>0.5:
    y_prediction.append(1)
  else:
    y_prediction.append(0)
y_prediction[:5]
```

Out[55]:

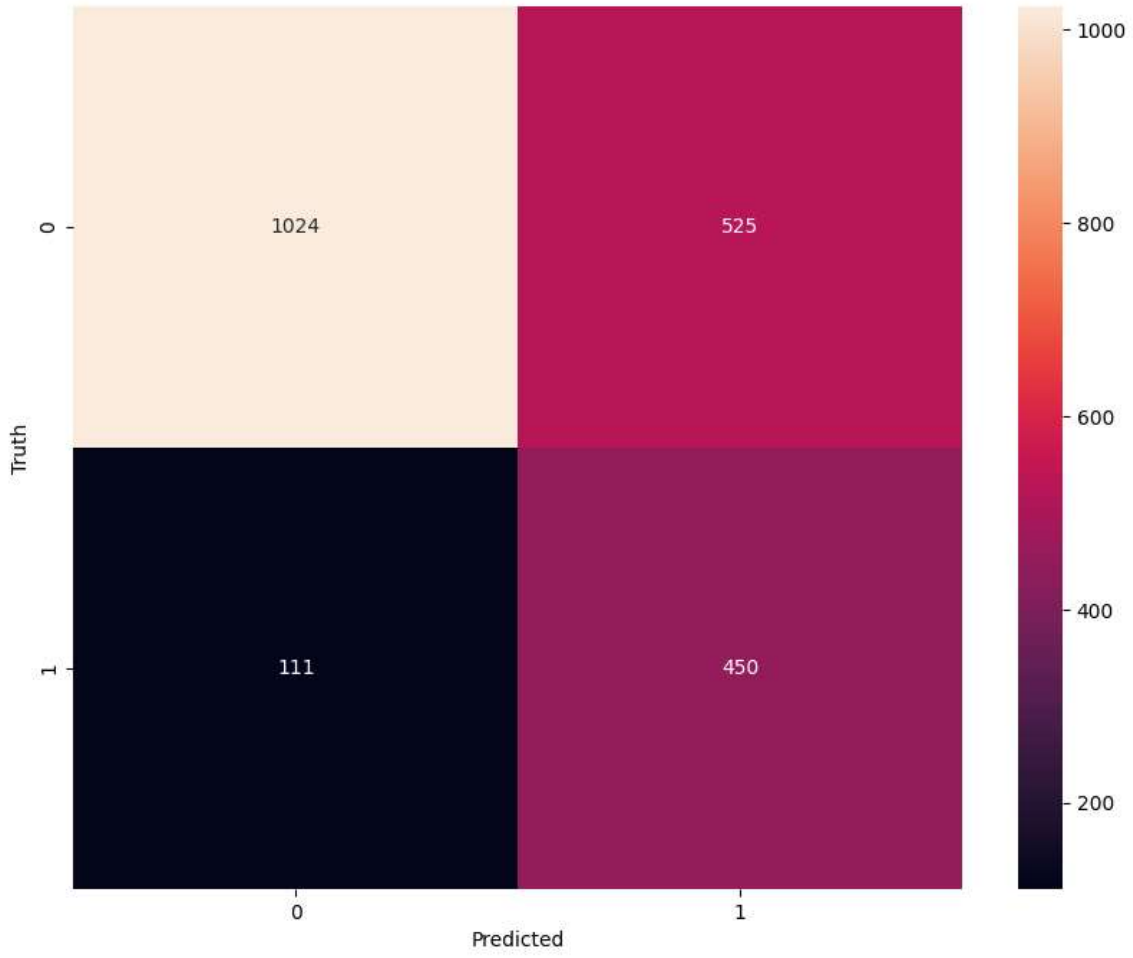
```
[0, 0, 1, 0, 0]
```

In [ ]:

```
cm=tf.math.confusion_matrix(labels=y_test,
                            predictions=y_prediction)
plt.figure(figsize=(10,8))
snp.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[61]:

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



Accuracy

In [ ]:

```
round((873+222)/(873+222+186+126),2)
```

Out[62]:

```
0.78
```

Precision for customers who did not churn

In [ ]:

```
round(873/(873+186),2)
```

Out[64]:

```
0.82
```

Precision for customers who actually

In [ ]:

```
round(222/(222+126),2)
```

Out[65]:

```
0.64
```

Recall for 0

In [ ]:

```
round(873/(873+126),2)
```

Out[66]:

```
0.87
```

Recall for 1

In [ ]:

```
round(222/(222+186),2)
```

Out[67]:

```
0.54
```