

```
In [34]: import pandas as pd
import numpy as np
df = pd.read_csv("WA_Fn-UseC_-Telco-Customer-Churn.csv")
df.head()
```

```
Out[34]:
```

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

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

```
Out[35]: (7043, 21)
```

```
In [36]: df.drop(["customerID"],axis=1,inplace=True)
df.head()
```

```
Out[36]:
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection
0	Female	0	Yes	No	1	No	No phone service	DSL	No	Yes	No
1	Male	0	No	No	34	Yes	No	DSL	Yes	No	Yes
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	No
3	Male	0	No	No	45	No	No phone service	DSL	Yes	No	Yes

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection
4	Female	0	No	No	2	Yes	No	Fiber optic	No	No	No

In [37]: `df.dtypes`

```
Out[37]: 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]: `df.TotalCharges.values`

```
Out[38]: array(['29.85', '1889.5', '108.15', ..., '346.45', '306.6', '6844.5'],
              dtype=object)
```

In [39]: `pd.to_numeric(df.TotalCharges,errors="coerce").isnull()`

```
Out[39]: 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 [40]: df[pd.to_numeric(df.TotalCharges,errors="coerce").isnull()]
```

```
Out[40]:
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtect
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 internet service	No internet 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 internet service	No internet 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 internet service	No internet service
3826	Male	0	Yes	Yes	0	Yes	Yes	No	No internet service	No internet service	No internet service
4380	Female	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service
5218	Male	0	Yes	Yes	0	Yes	No	No	No internet service	No internet service	No internet service
6670	Female	0	Yes	Yes	0	Yes	Yes	DSL	No	Yes	
6754	Male	0	No	Yes	0	Yes	Yes	DSL	Yes	Yes	

```
In [41]: df.iloc[488,-2]
```

```
Out[41]:
```

```
In [42]: df_1 = df[df.TotalCharges != " "]
df_1.shape
```

```
Out[42]: (7032, 20)
```

```
In [43]: df_1.TotalCharges = pd.to_numeric(df_1.TotalCharges)
```

C:\Users\U.R Computer\anaconda\lib\site-packages\pandas\core\generic.py:5168: 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

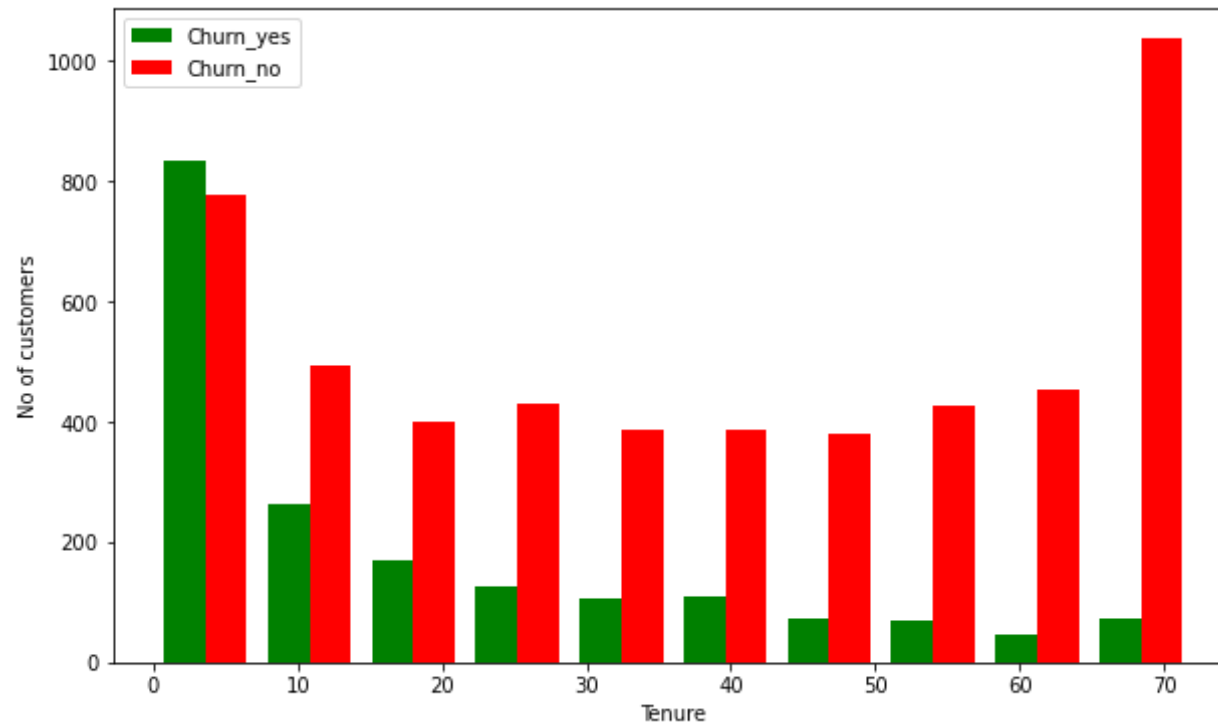
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
self[name] = value

```
In [44]: df_1.TotalCharges.dtype
```

```
Out[44]: dtype('float64')
```

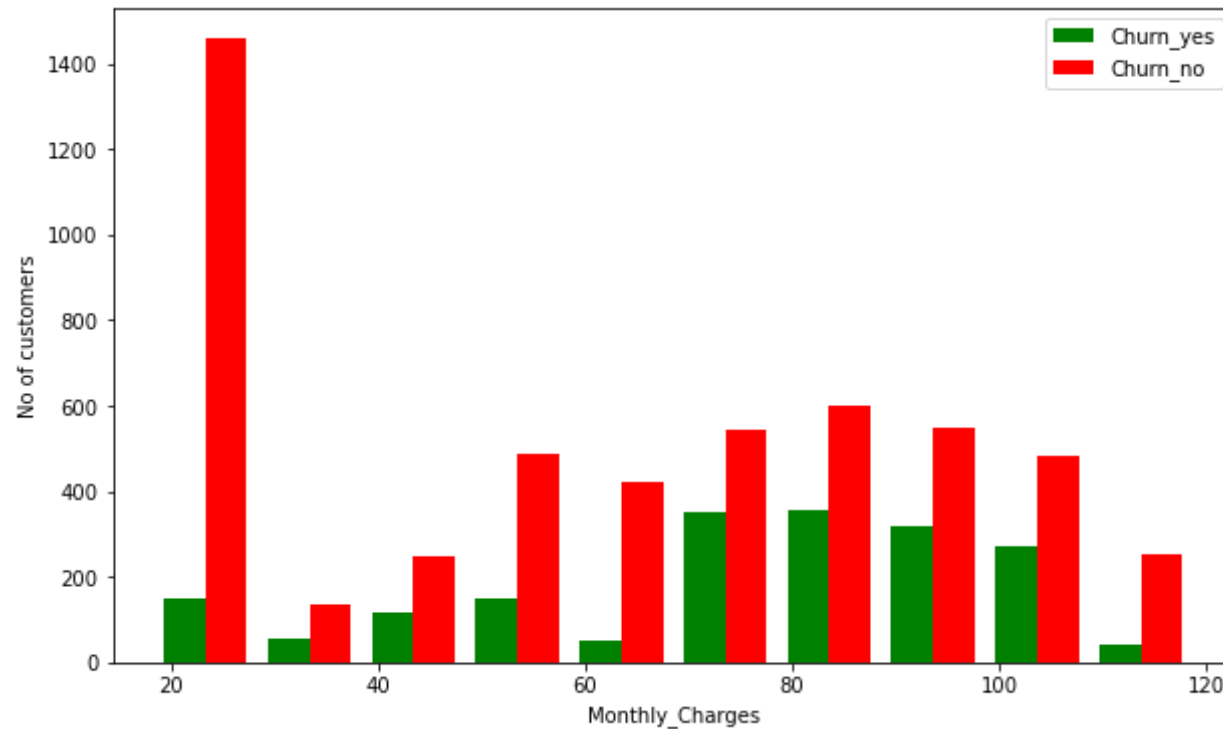
```
In [45]: import matplotlib.pyplot as plt
tenure_no = df[df.Churn=="No"].tenure
tenure_yes = df[df.Churn=="Yes"].tenure
plt.figure(figsize=(10,6))
plt.xlabel("Tenure")
plt.ylabel("No of customers")
plt.hist([tenure_yes,tenure_no],label=["Churn_yes","Churn_no"],color=["green","red"])
plt.legend()
```

```
Out[45]: <matplotlib.legend.Legend at 0x1f06c9ad0d0>
```



```
In [46]: Mc_no = df[df.Churn=="No"].MonthlyCharges
Mc_yes = df[df.Churn=="Yes"].MonthlyCharges
plt.figure(figsize=(10,6))
plt.xlabel("Monthly_Charges")
plt.ylabel("No of customers")
plt.hist([Mc_yes,Mc_no],label=["Churn_yes","Churn_no"],color=["green","red"])
plt.legend()
```

```
Out[46]: <matplotlib.legend.Legend at 0x1f06d002d90>
```



```
In [47]: def unique_col_val(df):
         for col in df:
             if df[col].dtype == "object":
                 print(f"{col} : {df[col].unique()}")
```

```
In [48]: unique_col_val(df_1)
```

```
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 [49]: df_1.replace("No internet service","No",inplace=True)
df_1.replace("No phone service","No",inplace=True)
```

C:\Users\U.R Computer\anaconda\lib\site-packages\pandas\core\frame.py:4379: 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#returning-a-view-versus-a-copy
return super().replace(

```
In [50]: unique_col_val(df_1)
```

```
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 [51]: df_1.columns
```

```
Out[51]: Index(['gender', 'SeniorCitizen', 'Partner', 'Dependents', 'tenure',
               'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
               'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
               'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod',
```

```
    'MonthlyCharges', 'TotalCharges', 'Churn'],
    dtype='object')
```

```
In [56]: yes_or_no = ['PhoneService', 'MultipleLines', 'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'St
for col in yes_or_no:
    df_1[col].replace({"Yes":1, "No":0}, inplace=True)
```


C:\Users\U.R Computer\anaconda\lib\site-packages\pandas\core\series.py:4563: 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#returning-a-view-versus-a-copy
return super().replace()

```
In [58]: df_1.head()
```

```
Out[58]:
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection
0	Female	0	1	0	1	0	0	DSL	0	1	0
1	Male	0	0	0	34	1	0	DSL	1	0	1
2	Male	0	0	0	2	1	0	DSL	1	1	0
3	Male	0	0	0	45	0	0	DSL	1	0	1
4	Female	0	0	0	2	1	0	Fiber optic	0	0	0



```
In [59]: df_1.gender.replace({"Female":1, "Male":0}, inplace=True)
```

C:\Users\U.R Computer\anaconda\lib\site-packages\pandas\core\series.py:4563: 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#returning-a-view-versus-a-copy
return super().replace()

```
In [62]: df_1.gender.unique()
```


Out[62]: array([1, 0], dtype=int64)

```
In [64]: df_2 = pd.get_dummies(data=df_1, columns=["InternetService","Contract","PaymentMethod"])
df_2.sample(4)
```

Out[64]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport
4921	1	0	1	1	39	1	1	0	0	0	
2991	1	1	1	0	37	1	0	0	0	1	
609	1	1	1	0	65	1	1	1	0	1	
5190	0	1	1	1	29	1	1	0	0	1	

```
In [65]: col_to_scale = ["tenure","MonthlyCharges","TotalCharges"]
from sklearn.preprocessing import MinMaxScaler
scale = MinMaxScaler()
df_2[col_to_scale] = scale.fit_transform(df_2[col_to_scale])
```

```
In [66]: df_2.sample(4)
```

Out[66]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport
4586	1	0	1	1	1.000000	1	1	1	1	1	
5496	0	0	1	1	0.591549	1	1	0	0	0	
4016	1	0	0	0	0.000000	1	0	0	0	0	
1747	1	0	1	1	0.028169	1	1	0	0	0	

```
In [75]: x = df_2.drop("Churn",axis=1)
y = df_2["Churn"]
x.sample(3)
```

Out[75]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport
--	--------	---------------	---------	------------	--------	--------------	---------------	----------------	--------------	------------------	-------------

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupp
7026	1	0	0	0	0.112676	1	0	0	0	0	
533	0	1	1	1	0.943662	1	1	0	0	1	
3703	0	0	0	1	0.112676	1	0	0	0	0	

```
In [77]: 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)
print(x_train.shape,x_test.shape)

(5625, 26) (1407, 26)
```

```
In [79]: import tensorflow as tf
from tensorflow import keras
```

```
In [80]: model = keras.models.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"]
)
```

```
In [83]: model.fit(x_train,y_train,epochs=32)

Epoch 1/32
176/176 [=====] - 0s 2ms/step - loss: 0.3740 - accuracy: 0.8217
Epoch 2/32
176/176 [=====] - 1s 3ms/step - loss: 0.3744 - accuracy: 0.8213
Epoch 3/32
176/176 [=====] - 1s 3ms/step - loss: 0.3744 - accuracy: 0.8203
Epoch 4/32
176/176 [=====] - 1s 4ms/step - loss: 0.3741 - accuracy: 0.8213
Epoch 5/32
176/176 [=====] - 1s 4ms/step - loss: 0.3735 - accuracy: 0.8236
Epoch 6/32
176/176 [=====] - 1s 5ms/step - loss: 0.3744 - accuracy: 0.8231
```

Epoch 7/32
176/176 [=====] - 1s 4ms/step - loss: 0.3733 - accuracy: 0.8226
Epoch 8/32
176/176 [=====] - 1s 4ms/step - loss: 0.3740 - accuracy: 0.8231
Epoch 9/32
176/176 [=====] - 1s 4ms/step - loss: 0.3732 - accuracy: 0.8233
Epoch 10/32
176/176 [=====] - 1s 4ms/step - loss: 0.3731 - accuracy: 0.8256
Epoch 11/32
176/176 [=====] - 1s 3ms/step - loss: 0.3732 - accuracy: 0.8217
Epoch 12/32
176/176 [=====] - 1s 4ms/step - loss: 0.3731 - accuracy: 0.8238
Epoch 13/32
176/176 [=====] - 1s 3ms/step - loss: 0.3731 - accuracy: 0.8213
Epoch 14/32
176/176 [=====] - 1s 3ms/step - loss: 0.3727 - accuracy: 0.8240
Epoch 15/32
176/176 [=====] - 1s 3ms/step - loss: 0.3726 - accuracy: 0.8236
Epoch 16/32
176/176 [=====] - 1s 4ms/step - loss: 0.3722 - accuracy: 0.8231
Epoch 17/32
176/176 [=====] - 1s 4ms/step - loss: 0.3727 - accuracy: 0.8222
Epoch 18/32
176/176 [=====] - 1s 4ms/step - loss: 0.3720 - accuracy: 0.8251
Epoch 19/32
176/176 [=====] - 1s 3ms/step - loss: 0.3722 - accuracy: 0.8219
Epoch 20/32
176/176 [=====] - 1s 6ms/step - loss: 0.3722 - accuracy: 0.8242
Epoch 21/32
176/176 [=====] - 1s 3ms/step - loss: 0.3725 - accuracy: 0.8215
Epoch 22/32
176/176 [=====] - 1s 3ms/step - loss: 0.3718 - accuracy: 0.8251
Epoch 23/32
176/176 [=====] - 1s 4ms/step - loss: 0.3714 - accuracy: 0.8251
Epoch 24/32
176/176 [=====] - 1s 5ms/step - loss: 0.3716 - accuracy: 0.8222
Epoch 25/32
176/176 [=====] - 1s 4ms/step - loss: 0.3716 - accuracy: 0.8233
Epoch 26/32
176/176 [=====] - 1s 4ms/step - loss: 0.3713 - accuracy: 0.8238
Epoch 27/32
176/176 [=====] - 1s 4ms/step - loss: 0.3714 - accuracy: 0.8233
Epoch 28/32
176/176 [=====] - 1s 4ms/step - loss: 0.3717 - accuracy: 0.8210
Epoch 29/32

```
176/176 [=====] - 1s 4ms/step - loss: 0.3720 - accuracy: 0.8249
Epoch 30/32
176/176 [=====] - 1s 4ms/step - loss: 0.3709 - accuracy: 0.8220
Epoch 31/32
176/176 [=====] - 1s 3ms/step - loss: 0.3709 - accuracy: 0.8235
Epoch 32/32
176/176 [=====] - 1s 4ms/step - loss: 0.3705 - accuracy: 0.8251
```

Out[83]: <tensorflow.python.keras.callbacks.History at 0x1f0780d81c0>

```
In [84]: model.evaluate(x_test,y_test)
```

```
44/44 [=====] - 0s 3ms/step - loss: 0.4493 - accuracy: 0.7918
```

Out[84]: [0.44934844970703125, 0.7917554974555969]

```
In [89]: y_pred = model.predict(x_test)
         y_pred[0:5]
```

Out[89]: array([[0.18165618],
 [0.57043386],
 [0.00751346],
 [0.77889264],
 [0.51461]], dtype=float32)

```
In [91]: y_test[0:10]
```

Out[91]: 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 [87]: y_pred_sigmoid = []
         for i in y_pred:
             if i > 0.5:
                 y_pred_sigmoid.append(1)
             else:
```

```
y_pred_sigmoid.append(0)
```

```
In [92]: y_pred_sigmoid[:10]
```

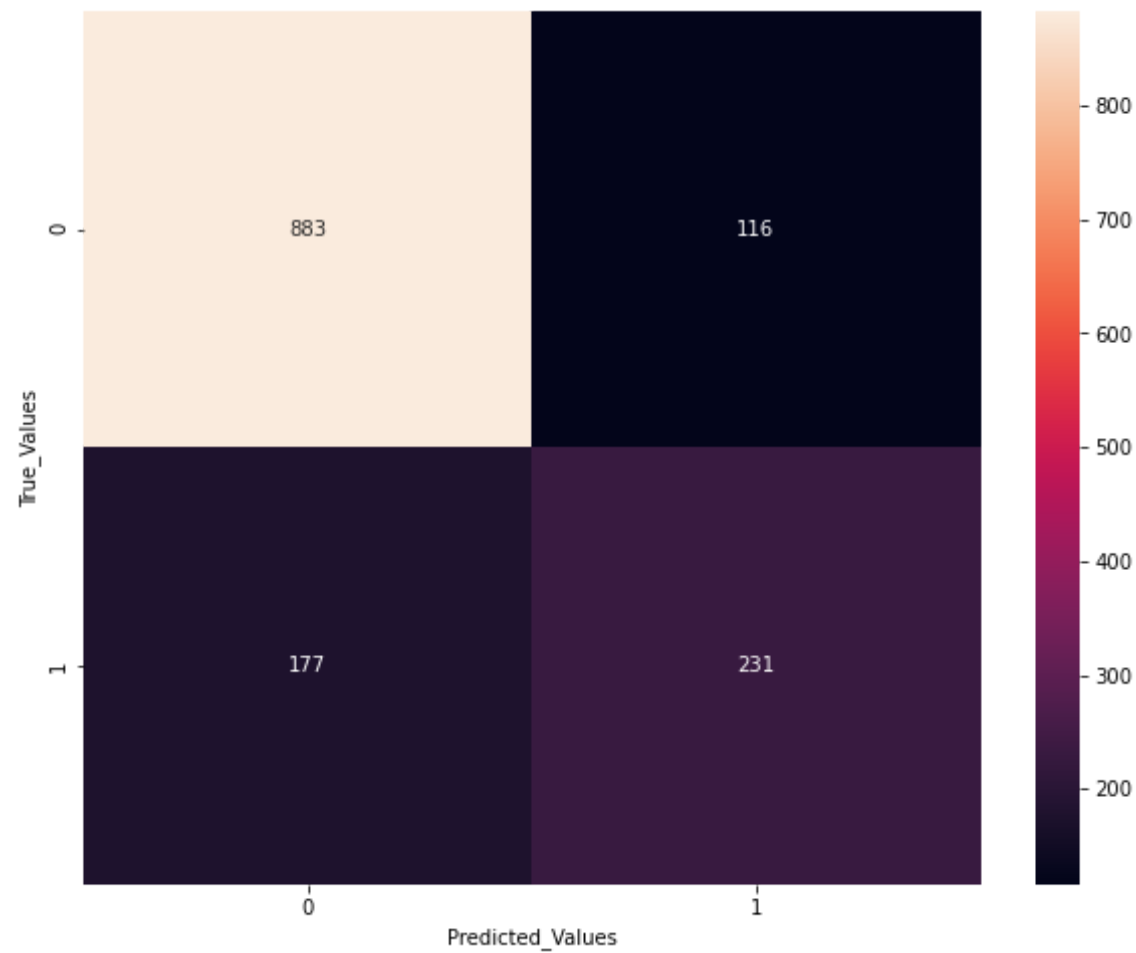
```
Out[92]: [0, 1, 0, 1, 1, 1, 0, 0, 0, 0]
```

```
In [93]: from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test,y_pred_sigmoid))
```

	precision	recall	f1-score	support
0	0.83	0.88	0.86	999
1	0.67	0.57	0.61	408
accuracy			0.79	1407
macro avg	0.75	0.73	0.73	1407
weighted avg	0.78	0.79	0.79	1407

```
In [101]: import seaborn as sns
cm = confusion_matrix(y_test,y_pred_sigmoid)
plt.figure(figsize=(10,8))
sns.heatmap(cm,annot=True,fmt="d")
plt.ylabel("True_Values")
plt.xlabel("Predicted_Values")
```

```
Out[101]: Text(0.5, 51.0, 'Predicted_Values')
```



In []: