Naan Mudhalvan Project

customer churn prediction

Team Members:

~Akshaya K- 2021115011

~Akshaya SM - 2021115012

~ Preetham V -2021115077

~Praveen M - 2021115075

~ Akash M - 2021115324

Naan Mudhalvan Project
customer churn prediction
Team Members:

~Akshaya K- 2021115011

~Akshaya SM - 2021115012

~ Preetham V -2021115077

~Praveen M - 2021115075

~ Akash M - 2021115324

```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a versio
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/telco-customer-churn/WA_Fn-UseC_-Telco-Customer-Churn.csv
```

Double-click (or enter) to edit

```
import matplotlib.pyplot as plt


df = pd.read_csv('/kaggle/input/telco-customer-churn/WA_Fn-UseC_-Telco-Customer-Churn.csv')
df.head(5)
```

	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

```
df.drop('customerID',axis='columns',inplace=True)
df.sample(5)
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetS
2128	Male	0	Yes	Yes	41	Yes	Yes	

```
df.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
dtype: object
```

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

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetS
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	

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

```
df1['TotalCharges'] = pd.to_numeric(df['TotalCharges'],errors='coerce')
```

```
/tmp/ipykernel_20/1694240200.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#returning-a-
  df1['TotalCharges'] = pd.to_numeric(df['TotalCharges'],errors='coerce')
```

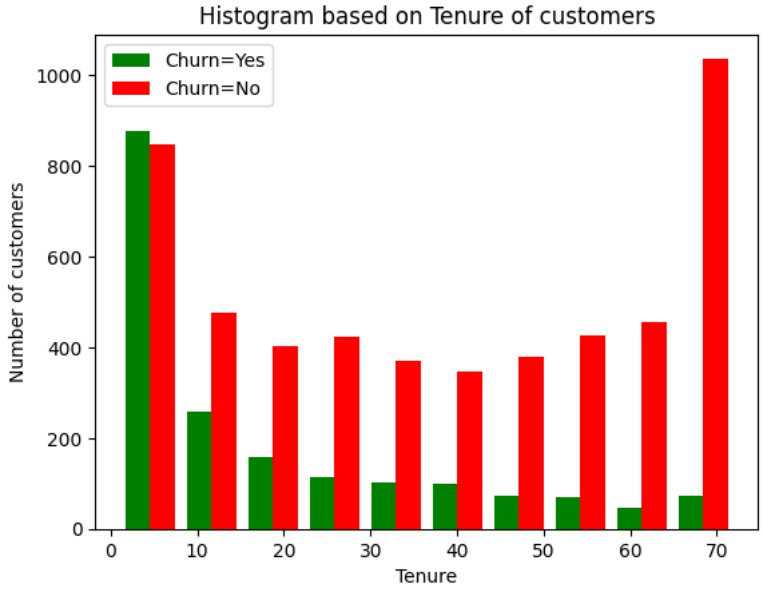
```
df1.dtypes
```

```
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        float64
Churn                object
dtype: object
```

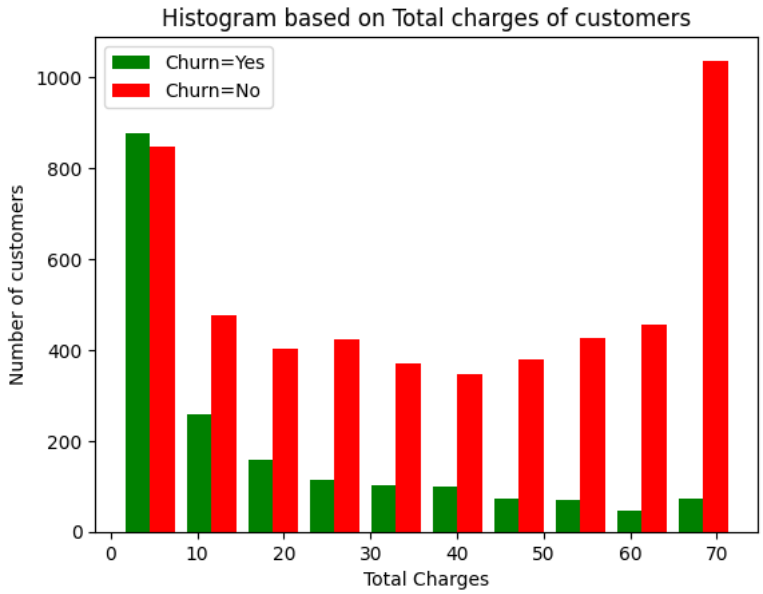
```
df_tenure_no = df1[df1.Churn == 'No'].tenure
df_tenure_yes = df1[df1.Churn == 'Yes'].tenure
plt.hist([df_tenure_yes,df_tenure_no],color=['green','red'],label=['Churn=Yes','Churn=No'])
plt.legend()
plt.xlabel('Tenure')
plt.ylabel('Number of customers')
plt.title('Histogram based on Tenure of customers')
```

```
Text(0.5, 1.0, 'Histogram based on Tenure of customers')
```




```
df_total_charges_no = df1[df1.Churn == 'No'].TotalCharges
df_total_charges_yes = df1[df1.Churn == 'Yes'].TotalCharges
plt.hist([df_tenure_yes,df_tenure_no],color=['green','red'],label=['Churn=Yes','Churn=No'])
plt.legend()
plt.xlabel('Total Charges')
plt.ylabel('Number of customers')
plt.title('Histogram based on Total charges of customers')
```

```
Text(0.5, 1.0, 'Histogram based on Total charges of customers')
```



```
def print_unique_values(df):
    for cols in df.columns:
        print(cols,df[cols].unique())


print_unique_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']
```

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

```
/tmp/ipykernel_20/628100714.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#returning-a-
  df1.replace('No phone service','No',inplace=True)
```

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

```
/tmp/ipykernel_20/4127402845.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#returning-a-
  df1.replace('No internet service','No',inplace=True)
```

```
print_unique_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']
```

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

```
df2.shape
```

```
(7032, 27)
```

```
print_unique_values(df2)
```

```
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']
OnlineSecurity ['No' 'Yes']
OnlineBackup ['Yes' 'No']
DeviceProtection ['No' 'Yes']
TechSupport ['No' 'Yes']
StreamingTV ['No' 'Yes']
StreamingMovies ['No' 'Yes']
PaperlessBilling ['Yes' 'No']
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']
InternetService_DSL [ True False]
InternetService_Fiber optic [False  True]
InternetService_No [False  True]
Contract_Month-to-month [ True False]
Contract_One year [False  True]
Contract_Two year [False  True]
PaymentMethod_Bank transfer (automatic) [False  True]
PaymentMethod_Credit card (automatic) [False  True]
PaymentMethod_Electronic check [ True False]
PaymentMethod_Mailed check [False  True]
```

```
df_true_false_cols = ['InternetService_DSL','InternetService_Fiber optic','InternetService_No','Contract_One year','Contract_
```

```
for cols in df_true_false_cols:
    df2[cols].replace({True:1,False:0},inplace=True)
```

```
print_unique_values(df2)
```

```
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']
OnlineSecurity ['No' 'Yes']
OnlineBackup ['Yes' 'No']
DeviceProtection ['No' 'Yes']
TechSupport ['No' 'Yes']
StreamingTV ['No' 'Yes']
StreamingMovies ['No' 'Yes']
PaperlessBilling ['Yes' 'No']
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']
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]
```

```
df_yes_no_cols = ['Partner','Dependents','PhoneService','MultipleLines','OnlineSecurity','OnlineBackup','DeviceProtection','T
for cols in df_yes_no_cols:
    df2[cols].replace({'Yes':1,'No':0},inplace=True)
print_unique_values(df2)
```

```
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]
OnlineSecurity [0 1]
```

```
OnlineBackup [1 0]
DeviceProtection [0 1]
TechSupport [0 1]
StreamingTV [0 1]
StreamingMovies [0 1]
PaperlessBilling [1 0]
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]
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]
```

```
df2['gender'].replace({'Female':1,'Male':0},inplace=True)
cols_to_scale = ['tenure','MonthlyCharges','TotalCharges']
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df2[cols_to_scale] = scaler.fit_transform(df2[cols_to_scale])
print_unique_values(df2)
```

```
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]
```

```
df2.dtypes
```

```
gender                                     int64
SeniorCitizen                              int64
Partner                                    int64
Dependents                                 int64
tenure                                   float64
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                        int64
InternetService_Fiber optic                int64
```

```
InternetService_No                        int64
Contract_Month-to-month                   int64
Contract_One year                         int64
Contract_Two year                         int64
PaymentMethod_Bank transfer (automatic)   int64
PaymentMethod_Credit card (automatic)     int64
PaymentMethod_Electronic check            int64
PaymentMethod_Mailed check                int64
dtype: object
```

```
X = df2.drop('Churn',axis='columns')
y = df2['Churn']
print('Shape of X:',X.shape)
print('Shape of y:',y.shape)
```

```
Shape of X: (7032, 26)
Shape of y: (7032,)
```

```
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('Shape of X_train:',X_train.shape)
print('Shape of X_test:',X_test.shape)
print('Shape of y_train:',y_train.shape)
print('Shape of y_test:',y_test.shape)
```

```
Shape of X_train: (5625, 26)
Shape of X_test: (1407, 26)
Shape of y_train: (5625,)
Shape of y_test: (1407,)
```

```
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=10)
```

```
Epoch 1/10
176/176 [==============================] - 1s 1ms/step - loss: 0.5006 - accuracy: 0.7550
Epoch 2/10
176/176 [==============================] - 0s 1ms/step - loss: 0.4357 - accuracy: 0.7902
Epoch 3/10
176/176 [==============================] - 0s 2ms/step - loss: 0.4244 - accuracy: 0.7977
Epoch 4/10
176/176 [==============================] - 0s 1ms/step - loss: 0.4196 - accuracy: 0.8021
Epoch 5/10
176/176 [==============================] - 0s 2ms/step - loss: 0.4162 - accuracy: 0.8043
Epoch 6/10
176/176 [==============================] - 0s 2ms/step - loss: 0.4148 - accuracy: 0.8048
Epoch 7/10
176/176 [==============================] - 0s 1ms/step - loss: 0.4128 - accuracy: 0.8066
Epoch 8/10
176/176 [==============================] - 0s 1ms/step - loss: 0.4121 - accuracy: 0.8076
Epoch 9/10
176/176 [==============================] - 0s 1ms/step - loss: 0.4110 - accuracy: 0.8071
Epoch 10/10
176/176 [==============================] - 0s 1ms/step - loss: 0.4097 - accuracy: 0.8089
<keras.callbacks.History at 0x7e596b5d7df0>
```

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

```
44/44 [==============================] - 0s 1ms/step - loss: 0.4416 - accuracy: 0.7889
[0.44159233570098877, 0.7889125943183899]
```

```
from sklearn.metrics import classification_report,confusion_matrix
yp = model.predict(X_test)
yp
```

```
44/44 [==============================] - 0s 1ms/step
array([[0.19795616],
       [0.3622312 ],
       [0.01491462],
       ...,
       [0.72733843],
       [0.6845046 ],
       [0.55635047]], dtype=float32)
```

```
yp.shape
```

```
(1407, 1)
```

```
yp[:5]
```

```
array([[0.19795616],
       [0.3622312 ],
       [0.01491462],
       [0.7472115 ],
       [0.48185456]], dtype=float32)
```

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

```
y_pred[:5]
```

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

```
y_test[:5]
```

```
2660    0
744     0
5579    1
64      1
3287    1
Name: Churn, dtype: int64
```

```
print('Classification Report:',classification_report(y_test,y_pred))
```

```
Classification Report:               precision    recall  f1-score   support

           0       0.84      0.87      0.85       999
           1       0.65      0.58      0.61       408

    accuracy                           0.79      1407
   macro avg       0.74      0.73      0.73      1407
weighted avg       0.78      0.79      0.79      1407
```

```
import seaborn as sns
cm = confusion_matrix(y_test,y_pred)
sns.heatmap(cm,annot=True,fmt='.2f')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
```

```
Text(50.722222222222214, 0.5, 'True')
```