```
import pandas as pd
import os
from skimage.transform import resize
from skimage.io import imread
import numpy as np
import matplotlib.pyplot as plt


Categories=['total_normal','total_pneumonia']


flat_data_arr=[]
target_arr=[]
flat_data_arr_test=[]
target_arr_test=[]


datadir = 'D:/github/archive/chest_xray/total_dataset'
datadir2 = 'chest_xray/test'


for i in Categories:

    print(f'loading... category : {i}')
    path=os.path.join(datadir,i)
    for img in os.listdir(path):
        img_array=imread(os.path.join(path,img))
        img_resized=resize(img_array,(150,150,3))
        flat_data_arr.append(img_resized.flatten())
        target_arr.append(Categories.index(i))
    print(f'loaded category:{i} successfully')
```

```
loading... category : total_normal
loaded category:total_normal successfully
loading... category : total_pneumonia
loaded category:total_pneumonia successfully
```

flat_data_test=np.array(flat_data_arr_test) target_test=np.array(target_arr_test) df=pd.DataFrame(flat_data_test) df['Target']=target_test x_test2=df.iloc[:,:-1] y_test2=df.iloc[:,-1]

```
flat_data=np.array(flat_data_arr)
target=np.array(target_arr)

df=pd.DataFrame(flat_data)
df['Target']=target

x=df.iloc[:,:-1]
y=df.iloc[:,-1]
x.shape
```

```
.
```

```
(5840, 67500)
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Ensure x_flat is a NumPy array with at least 2 dimensions
x_flat = np.array(x if x.ndim == 2 else x.reshape(x.shape[0], -1))

# Number of features to plot
num_features_to_plot = x_flat.shape[1]  # Plot up to 5 features or fewer if less are avai

# Plotting
plt.figure(figsize=(12, 6))
for i in range(num_features_to_plot):
    plt.subplot(1, num_features_to_plot, i + 1)
    sns.histplot(x_flat[:, i], kde=True)  # KDE shows if data is close to a normal distri
    plt.title(f"Feature {i + 1}")
    plt.xlabel("Value")
    plt.ylabel("Frequency")

plt.tight_layout()
plt.show()
```

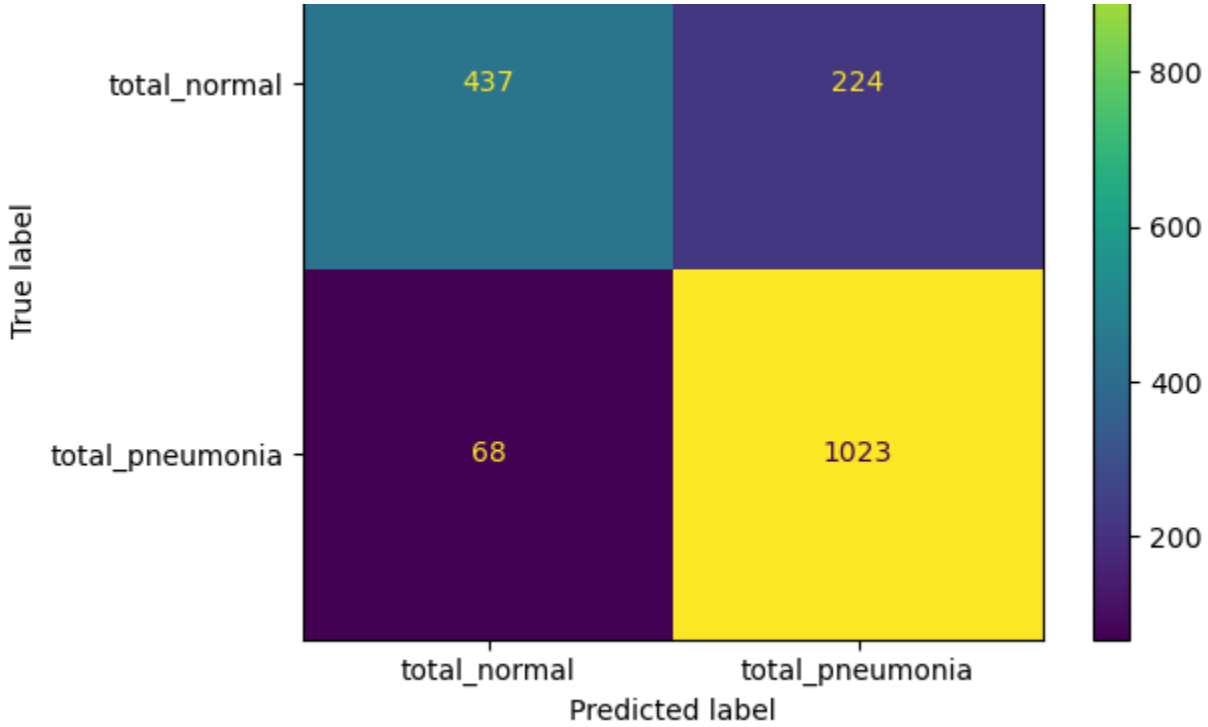
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
```

```
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import GaussianNB, ComplementNB, MultinomialNB
from sklearn.metrics import (
    accuracy_score,
    confusion_matrix,
    ConfusionMatrixDisplay,
    f1_score,
)
```

```
# -------------------- test data used here is made using split -----------------------
```

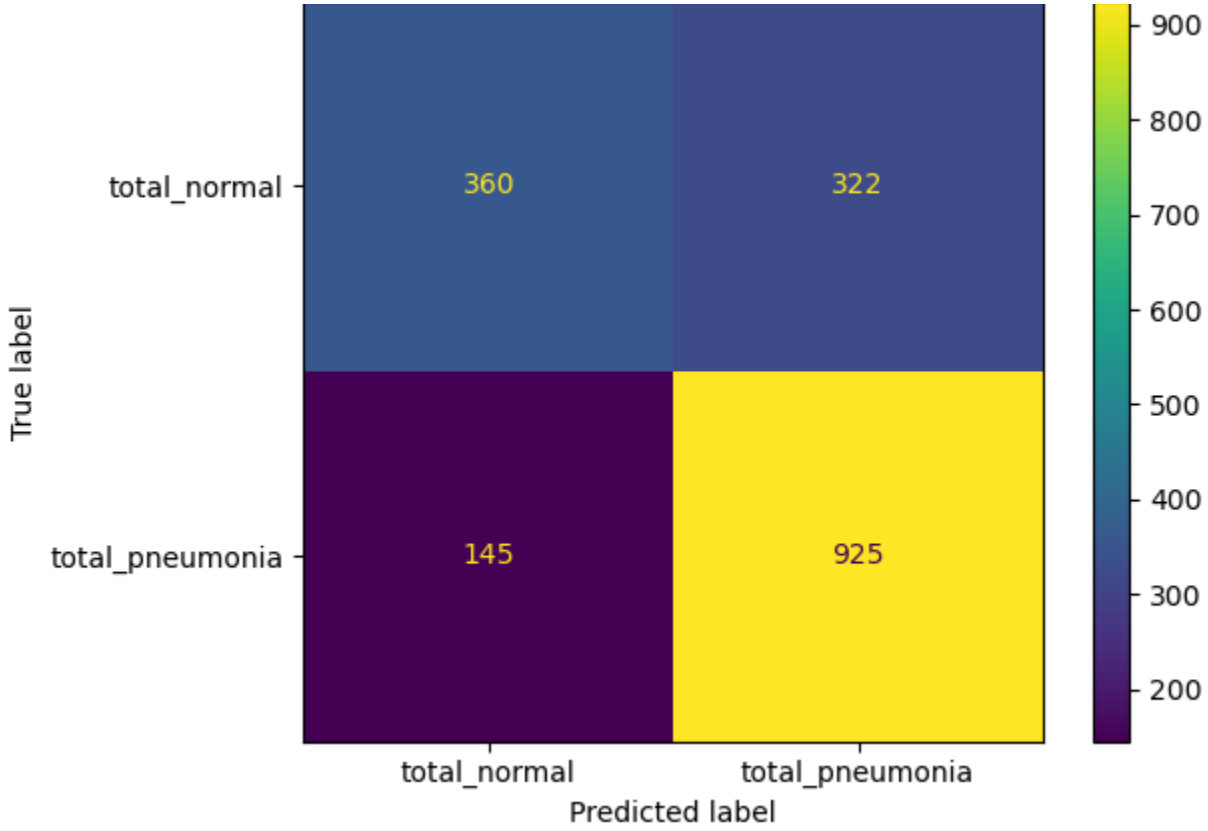
```
classifer_Gauss = GaussianNB()
classifer_Ber = BernoulliNB()
classifer_Comp = ComplementNB()
classifer_Multi = MultinomialNB()
model_Gauss = classifer_Gauss.fit(x_train, y_train)
model_Ber = classifer_Ber.fit(x_train, y_train)
model_Comp = classifer_Comp.fit(x_train, y_train)
model_Multi = classifer_Multi.fit(x_train, y_train)


from sklearn.metrics import classification_report


# for Gaussian NB


y_pred=model_Gauss.predict(x_test)
print("The predicted Data is :")
print(y_pred)
print("The actual data is:")
print(np.array(y_test))
print(f"The model is {accuracy_score(y_pred,y_test)*100}% accurate")
print(classification_report(y_test, y_pred, target_names=Categories))
cm = confusion_matrix(y_pred, y_test)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=Categories)
disp.plot()
model_Gauss.score(x, y)
```

```
The predicted Data is :
[1 0 1 ... 1 0 1]
The actual data is:
[1 0 0 ... 1 0 1]
The model is 83.33333333333334% accurate
                precision    recall  f1-score   support

   total_normal       0.66      0.87      0.75       505
total_pneumonia       0.94      0.82      0.88      1247

       accuracy                           0.83      1752
      macro avg       0.80      0.84      0.81      1752
   weighted avg       0.86      0.83      0.84      1752

0.8287671232876712
```




```
# for Bernoulli NB


y_pred=model_Ber.predict(x_test)
print("The predicted Data is :")
print(y_pred)
print("The actual data is:")
print(np.array(y_test))
print(f"The model is {accuracy_score(y_pred,y_test)*100}% accurate")
print(classification_report(y_test, y_pred, target_names=Categories))
cm = confusion_matrix(y_pred, y_test)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=Categories)
disp.plot()
```

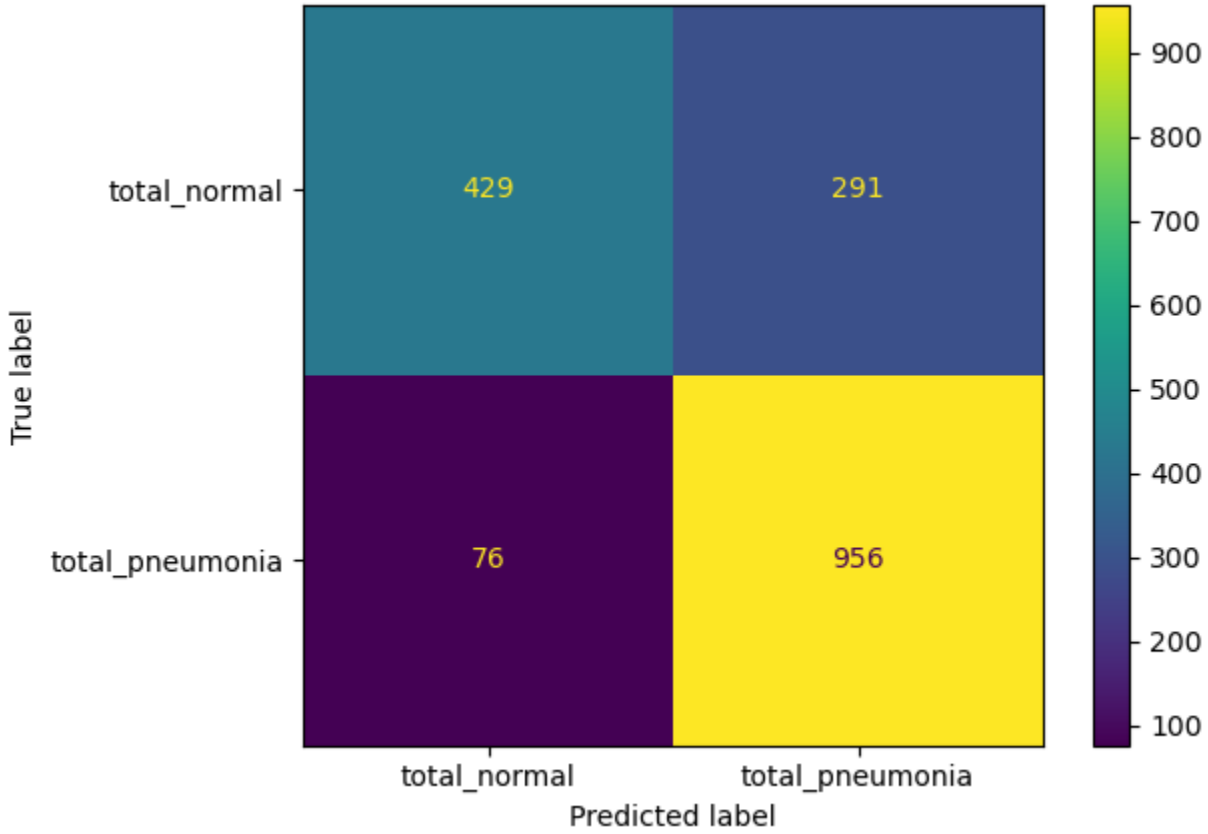
```
The predicted Data is :
[1 1 1 ... 1 0 1]
The actual data is:
[1 0 0 ... 1 0 1]
The model is 73.34474885844749% accurate
                precision    recall  f1-score   support

   total_normal       0.53      0.71      0.61       505
total_pneumonia       0.86      0.74      0.80      1247

       accuracy                           0.73      1752
      macro avg       0.70      0.73      0.70      1752
   weighted avg       0.77      0.73      0.74      1752

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x25c06a546a0>
```




```
# for Complement NB


y_pred=model_Comp.predict(x_test)
print("The predicted Data is :")
print(y_pred)
print("The actual data is:")
print(np.array(y_test))
print(f"The model is {accuracy_score(y_pred,y_test)*100}% accurate")
print(classification_report(y_test, y_pred, target_names=Categories))
cm = confusion_matrix(y_pred, y_test)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=Categories)
disp.plot()
```

```
The predicted Data is :
[1 0 1 ... 1 0 1]
The actual data is:
[1 0 0 ... 1 0 1]
The model is 79.05251141552512% accurate
                precision    recall  f1-score   support

   total_normal       0.60      0.85      0.70       505
total_pneumonia       0.93      0.77      0.84      1247

       accuracy                           0.79      1752
      macro avg       0.76      0.81      0.77      1752
   weighted avg       0.83      0.79      0.80      1752
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x25c067f66d0>
```

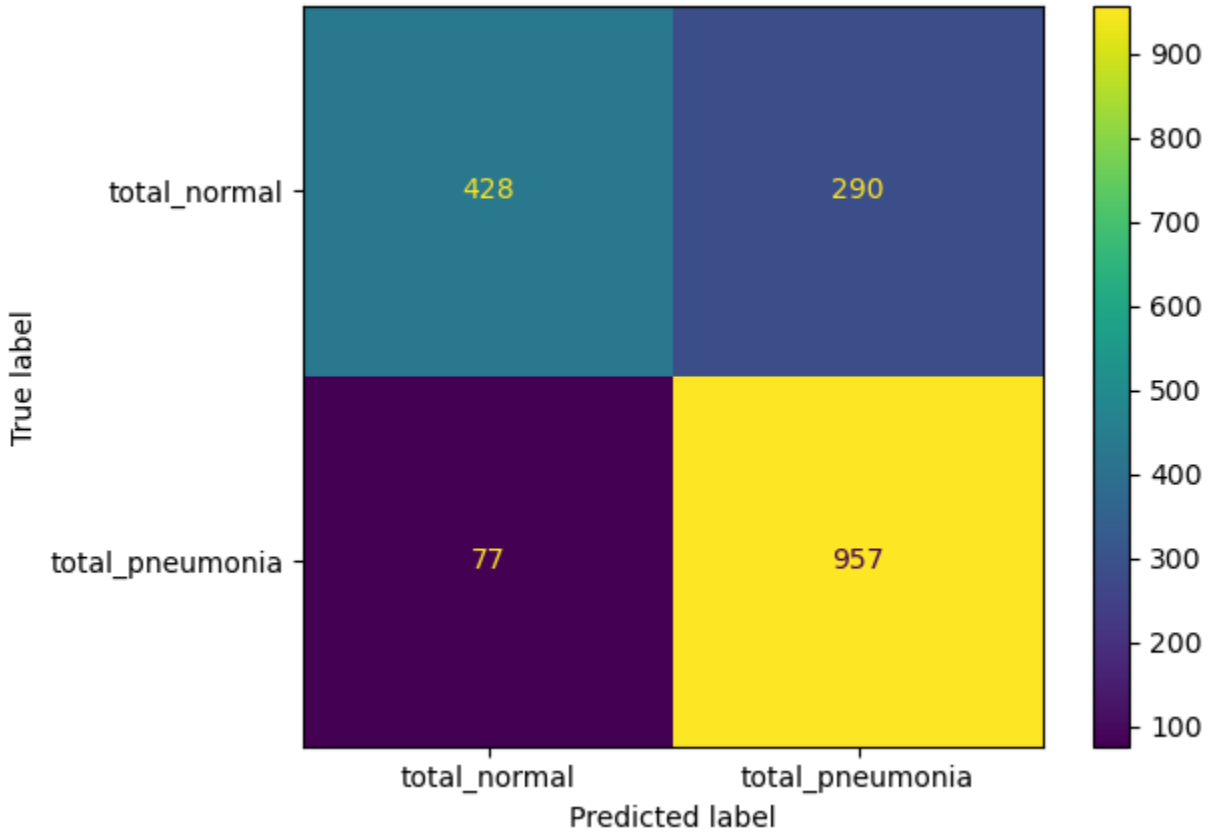



```
# for Multinomial NB


y_pred=model_Multi.predict(x_test)
print("The predicted Data is :")
print(y_pred)
print("The actual data is:")
print(np.array(y_test))
print(f"The model is {accuracy_score(y_pred,y_test)*100}% accurate")
print(classification_report(y_test, y_pred, target_names=Categories))
cm = confusion_matrix(y_pred, y_test)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=Categories)
disp.plot()
```

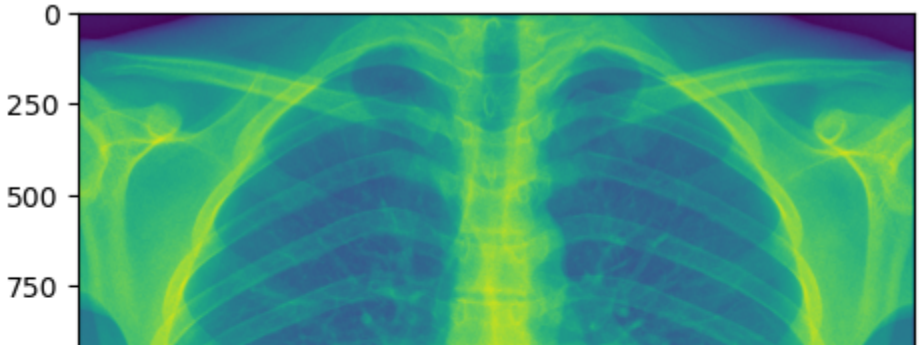
```
The predicted Data is :
[1 0 1 ... 1 0 1]
The actual data is:
[1 0 0 ... 1 0 1]
The model is 79.05251141552512% accurate
                precision    recall  f1-score   support

   total_normal       0.60      0.85      0.70       505
total_pneumonia       0.93      0.77      0.84      1247
```

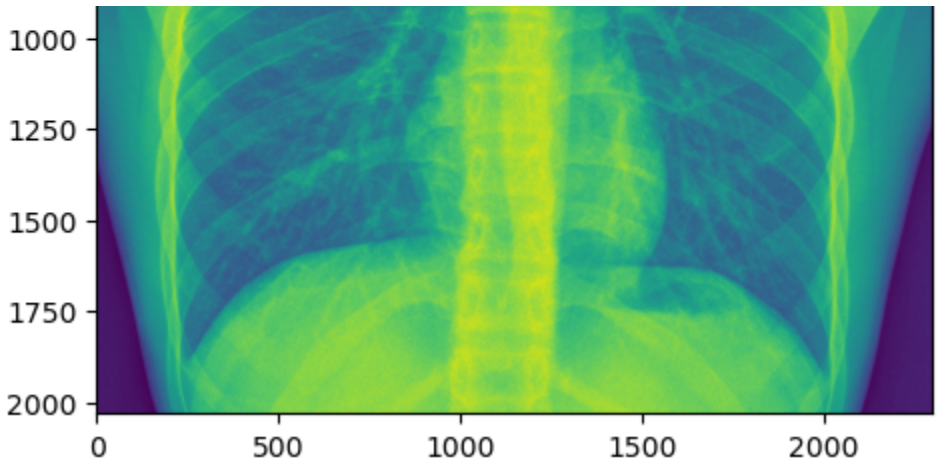
```
              accuracy                           0.79      1752
             macro avg       0.76      0.81      0.77      1752
          weighted avg       0.83      0.79      0.80      1752
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x25c1667a610>
```



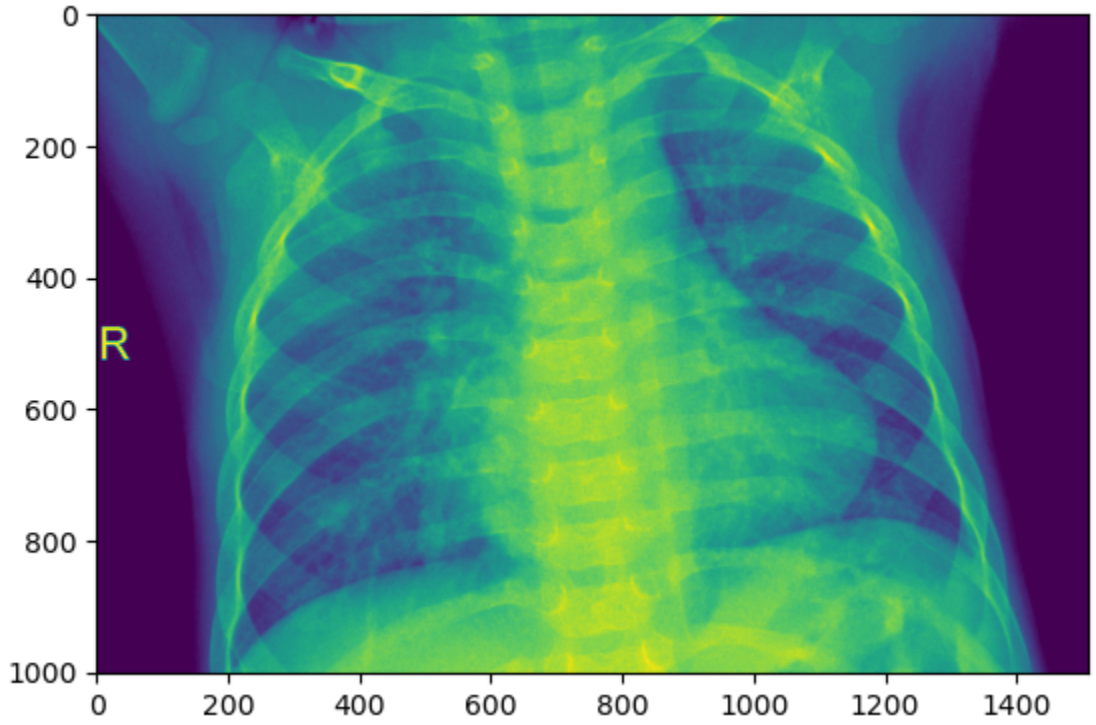

```
url=input('Enter URL of Image :')
img=imread(url)
plt.imshow(img)
plt.show()
img_resize=resize(img,(150,150,3))
l=[img_resize.flatten()]
probability=model_Gauss.predict_proba(l)
for ind,val in enumerate(Categories):
    print(f'{val} = {probability[0][ind]*100}%')
print("The predicted image is : "+Categories[model_Gauss.predict(l)[0]])
```






```
total_normal = 100.0%
total_pneumonia = 0.0%
The predicted image is : total_normal
```

```
url=input('Enter URL of Image :')
img=imread(url)
plt.imshow(img)
plt.show()
img_resize=resize(img,(150,150,3))
l=[img_resize.flatten()]
probability=model_Gauss.predict_proba(l)
for ind,val in enumerate(Categories):
    print(f'{val} = {probability[0][ind]*100}%')
print("The predicted image is : "+Categories[model_Gauss.predict(l)[0]])
```




```
total_normal = 0.0%
total_pneumonia = 100.0%
```

```
The predicted image is : total_pneumonia
```

```
# ----------------------------------------------------------------------------------------
```

```
# -------------------- test data used here is the actual data in the test directory -----

y_pred2=model_Gauss.predict(x_test2)
#print("The predicted Data is :")
#print(y_pred2)
#print("The actual data is:")
#print(np.array(y_test2))
print(f"The model is {accuracy_score(y_pred2,y_test2)*100}% accurate")
```

```
The model is 71.7948717948718% accurate
```

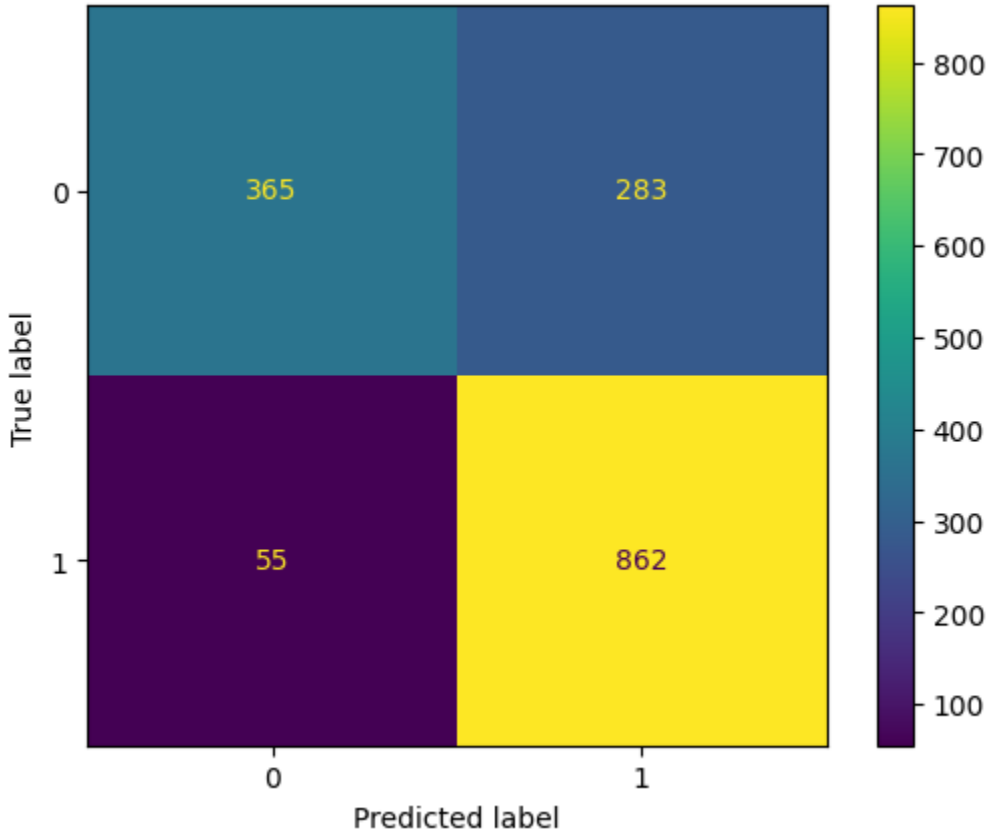
```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred, target_names=['Normal', 'Viral Pneumonia']))
print(classification_report(y_test2, y_pred2, target_names=['Normal', 'Viral Pneumonia'])
```

```
                    precision    recall  f1-score   support

            Normal       0.56      0.87      0.68       420
     Viral Pneumonia       0.94      0.75      0.84      1145

          accuracy                           0.78      1565
         macro avg       0.75      0.81      0.76      1565
      weighted avg       0.84      0.78      0.80      1565
```

```
weighted avg       0.84      0.78      0.80      1565

                precision    recall  f1-score   support

        Normal       0.62      0.63      0.63       234
Viral Pneumonia       0.78      0.77      0.77       390

      accuracy                           0.72       624
     macro avg       0.70      0.70      0.70       624
  weighted avg       0.72      0.72      0.72       624
```

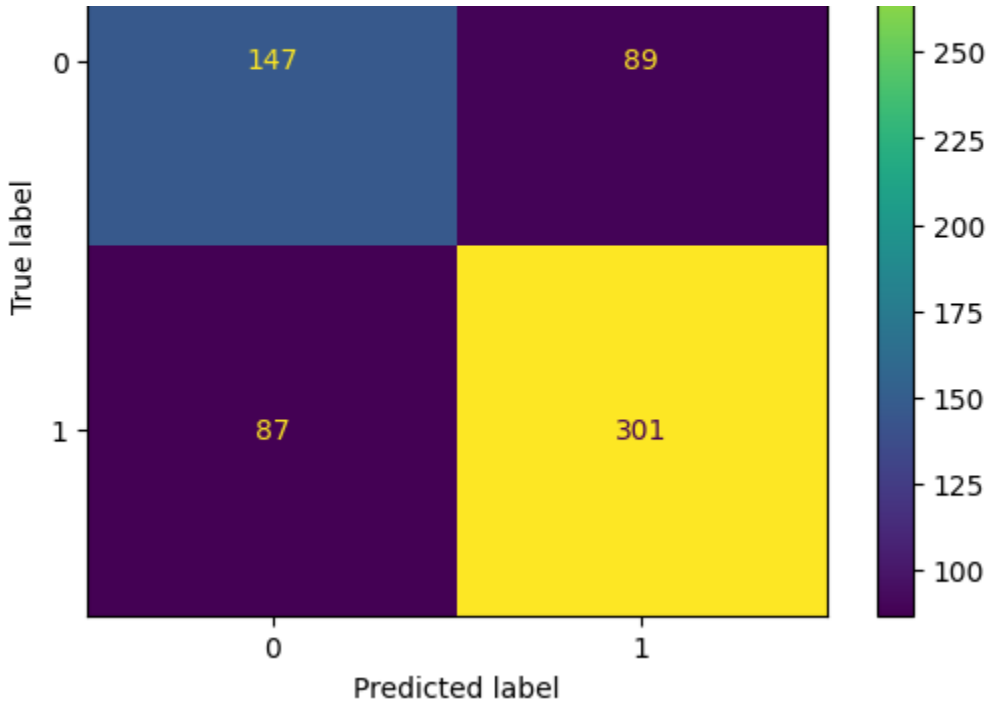
```
cm = confusion_matrix(y_pred, y_test)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classifer.classes_)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1ab603b6c90>
```




```
cm = confusion_matrix(y_pred2, y_test2)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=classifer.classes_)
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x1ab60377440>
```




```
url=input('Enter URL of Image :')
img=imread(url)
plt.imshow(img)
plt.show()
img_resize=resize(img,(150,150,3))
l=[img_resize.flatten()]
probability=model.predict_proba(l)
for ind,val in enumerate(Categories):
    print(f'{val} = {probability[0][ind]*100}%')
print("The predicted image is : "+Categories[model.predict(l)[0]])
```

```
url=input('Enter URL of Image :')
img=imread(url)
plt.imshow(img)
plt.show()
img_resize=resize(img,(150,150,3))
l=[img_resize.flatten()]
probability=model.predict_proba(l)
for ind,val in enumerate(Categories):
    print(f'{val} = {probability[0][ind]*100}%')
print("The predicted image is : "+Categories[model.predict(l)[0]])
```