```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set(style="whitegrid")
import os
import glob as gb
import cv2
import tensorflow as tf
import keras


# On indique ici que l'on utilise PyDrive et on importe les modules nécessaires
!pip install -U -q PyDrive
import os
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

# Authentification et création du client PyDrive
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)

# Choix du répertoire qui va contenir les fichiers téléchargés
dossier = os.path.expanduser('/content/Deap_Learning/data/TEST/hummingbird')
try:
  os.makedirs(dossier)
except: pass

# Parcours du dossier Google Drive et téléchargement des fichiers
liste_fichiers = drive.ListFile(
    {'q': "'15rqy5-h6T-jZcFFV9gLACyHeSxxOjKyJ' in parents"}).GetList()

for fichier in liste_fichiers:
  nom = os.path.join(dossier, fichier['title'])
  print('Téléchargement de {}'.format(nom))
  nouveauFichier = drive.CreateFile({'id': fichier['id']})
  nouveauFichier.GetContentFile(nom)

# Choix du répertoire qui va contenir les fichiers téléchargés
dossier = os.path.expanduser('/content/Deap_Learning/data/TEST/koala')
try:
  os.makedirs(dossier)
except: pass

# Parcours du dossier Google Drive et téléchargement des fichiers
liste_fichiers = drive.ListFile(
    {'q': "'1L7ujWWoVABR6F4sdxVquIMl8KwVKmckQ' in parents"}).GetList()

for fichier in liste_fichiers:
  nom = os.path.join(dossier, fichier['title'])
  print('Téléchargement de {}'.format(nom))
  nouveauFichier = drive.CreateFile({'id': fichier['id']})
  nouveauFichier.GetContentFile(nom)

# Choix du répertoire qui va contenir les fichiers téléchargés
dossier = os.path.expanduser('/content/Deap_Learning/data/TRAIN/hummingbird')
try:
  os.makedirs(dossier)
except: pass

# Parcours du dossier Google Drive et téléchargement des fichiers
liste_fichiers = drive.ListFile(
    {'q': "'1LcKyr_uybwy78tEB2aPEZol87CzVg-S1' in parents"}).GetList()

for fichier in liste_fichiers:
  nom = os.path.join(dossier, fichier['title'])
  print('Téléchargement de {}'.format(nom))
  nouveauFichier = drive.CreateFile({'id': fichier['id']})
  nouveauFichier.GetContentFile(nom)

# Choix du répertoire qui va contenir les fichiers téléchargés
dossier = os.path.expanduser('/content/Deap_Learning/data/TRAIN/koala')
try:
  os.makedirs(dossier)
except: pass

# Parcours du dossier Google Drive et téléchargement des fichiers
```

```
# Parcours du dossier Google Drive et telechargement des fichiers
liste_fichiers = drive.ListFile(
    {'q': "'1dgJFqmtQyUkRh2poKGAbI5wkKRqKyflr' in parents"}).GetList()

for fichier in liste_fichiers:
  nom = os.path.join(dossier, fichier['title'])
  print('Téléchargement de {}'.format(nom))
  nouveauFichier = drive.CreateFile({'id': fichier['id']})
  nouveauFichier.GetContentFile(nom)
```

```
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/7d59076173.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/79dc8956bf.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/9bf7dd7895.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/7ebb13cd63.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/9ca7babf11.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/6ce9e3a265.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/9f8451e1c5.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/8d2f165204.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/85a6803bf0.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/7d5fa491c6.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/77ae6058e2.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/7c6b58ad80.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/8ef90bdb03.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/5a2d678016.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/3e710d8da6.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/3b58222325.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/42c714ccb0.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/6a1cd7b301.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/53e6ef5129.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/1f3883417c.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/3d09dc2987.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/6c55ea6cca.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/1a9acbddfd.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/27bcdbd57f.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/57dd0f6d03.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/30be5547ce.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/5fbc3e3756.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/5a3251eb72.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/3d5e48ca8b.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/2aae24b29d.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/5ba26c9060.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/2f07008106.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/33efc25c32.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/30fa42ded0.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/3d52baf740.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/1fdc95801b.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/51fb5f79ee.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/3f1b884203.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/4db506f55b.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/3b032f05b1.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/3c6d6206ec.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/2e8fe24ad3.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/3f4b2076a8.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/4c78791147.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/52a9ac3802.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/1a76c60c40.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/28a37c134f.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/29d5fb0144.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/34d06d0ded.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/0d02c41ab5.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/063ba52395.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/14be31fe67.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/0e8eb6b2e9.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/0042f2effe.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/11fec2920e.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/09d035c85d.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/1a660142ae.jpg
Téléchargement de /content/Deap_Learning/data/TRAIN/koala/0eb924dd00.jpg
```

Présentation du dataset

```
### for Jupyter
trainpath = '/content/Deap_Learning/data/TRAIN/'
testpath = '/content/Deap_Learning/data/TEST/'


for folder in  os.listdir(trainpath) :
    files = gb.glob(pathname= str( trainpath + folder + '/*.jpg'))
    print(f'For training data , found {len(files)} in folder {folder}')
```

```
 For training data , found 60 in folder koala
 For training data , found 60 in folder hummingbird
```

```
for folder in  os.listdir(testpath) :
    files = gb.glob(pathname= str( testpath + folder + '/*.jpg'))
    print(f'For testing data , found {len(files)} in folder {folder}')
```

```
 For testing data , found 30 in folder koala
 For testing data , found 30 in folder hummingbird
```

```
code = {'hummingbird':0 ,'koala':1}

def getcode(n) :
    for x , y in code.items() :
        if n == y :
            return x
```

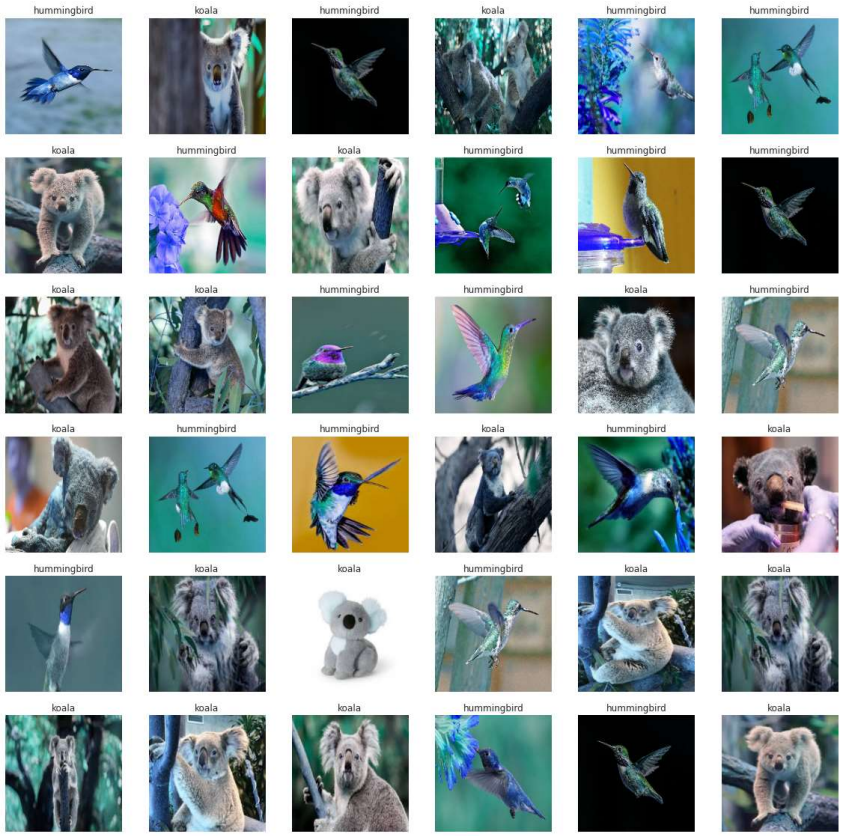
```
size = []
for folder in  os.listdir(trainpath) :
    files = gb.glob(pathname= str( trainpath + folder + '/*.jpg'))
    for file in files:
        image = plt.imread(file)
        size.append(image.shape)
pd.Series(size).value_counts()
```

```
 (183, 275, 3)     13
 (168, 300, 3)      6
 (194, 259, 3)      6
 (225, 225, 3)      4
 (162, 311, 3)      4
                   ..
 (174, 290, 3)      1
 (579, 1028, 3)     1
 (182, 276, 3)      1
 (346, 568, 3)      1
 (821, 1024, 3)     1
 Length: 88, dtype: int64
```

```
s = 170

X_train = []
y_train = []
for folder in  os.listdir(trainpath) :
    files = gb.glob(pathname= str( trainpath + folder + '/*.jpg'))
    for file in files:
        image = cv2.imread(file)
        image_array = cv2.resize(image , (s,s))
        X_train.append(list(image_array))
        y_train.append(code[folder])

print(f'we have {len(X_train)} items in X_train')
```

```
 we have 120 items in X_train
```

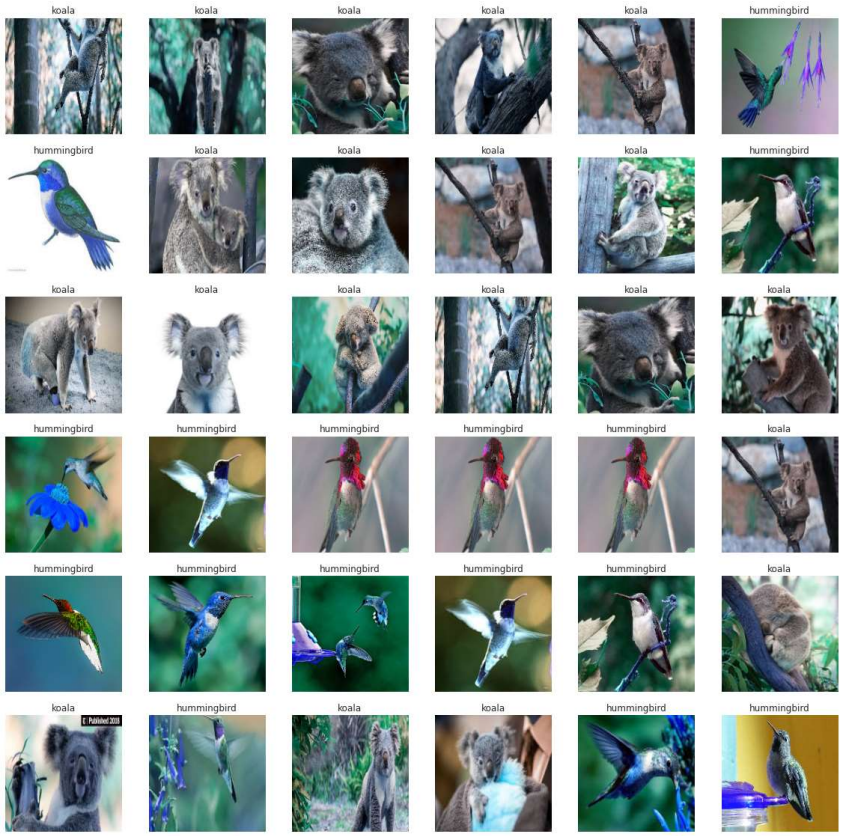
```
plt.figure(figsize=(20,20))
for n , i in enumerate(list(np.random.randint(0,len(X_train),36))) :
    plt.subplot(6,6,n+1)
    plt.imshow(X_train[i])
    plt.axis('off')
    plt.title(getcode(y_train[i]))
```




```
s = 170


X_test = []
y_test = []
for folder in  os.listdir(testpath) :
    files = gb.glob(pathname= str( testpath + folder + '/*.jpg'))
    for file in files:
        image = cv2.imread(file)
        image_array = cv2.resize(image , (s,s))
        X_test.append(list(image_array))
        y_test.append(code[folder])


print(f'we have {len(X_test)} items in X_test')
```

```
we have 60 items in X_test
```

```
plt.figure(figsize=(20,20))
for n , i in enumerate(list(np.random.randint(0,len(X_test),36))) :
    plt.subplot(6,6,n+1)
    plt.imshow(X_test[i])
    plt.axis('off')
    plt.title(getcode(y_test[i]))
```




```
X_train = np.array(X_train)
X_test = np.array(X_test)
y_train = np.array(y_train)
y_test = np.array(y_test)

print(f'X_train shape  is {X_train.shape}')
print(f'X_test shape  is {X_test.shape}')
print(f'y_train shape  is {y_train.shape}')
print(f'y_test shape  is {y_test.shape}')
```

```
X_train shape  is (120, 170, 170, 3)
X_test shape  is (60, 170, 170, 3)
y_train shape  is (120,)
y_test shape  is (60,)
```

Architecture

```
KerasModel = keras.models.Sequential([
        keras.layers.Conv2D(20,kernel_size=(3,3),activation='relu',input_shape=(s,s,3)),
        keras.layers.Conv2D(15,kernel_size=(3,3),activation='relu'),
        keras.layers.MaxPool2D(4,4),
        keras.layers.Flatten() ,
        keras.layers.Dense(50,activation='relu') ,
        keras.layers.Dropout(rate=0.5) ,
        keras.layers.Dense(2,activation='softmax') ,
        ])

KerasModel.compile(optimizer ='adam',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
```

Adam est l'optimiseur choisi, sparse_categorical_crossentropy correspond à la fonction de perte à minimiser lors de l'entraînement. metrics=['accuracy'] spécifie les métriques à évaluer pendant l'entraînement et les tests. Dans ce cas, la seule métrique spécifiée est l'accuracy (précision), qui est le pourcentage de prédictions correctes pour les échantillons d'entraînement et de test.

```
print('Model Details are : ')
print(KerasModel.summary())
```

```
Model Details are :
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 168, 168, 20)      560

 conv2d_1 (Conv2D)           (None, 166, 166, 15)      2715

 max_pooling2d (MaxPooling2D  (None, 41, 41, 15)       0
 )

 flatten (Flatten)           (None, 25215)             0

 dense (Dense)               (None, 50)                1260800

 dropout (Dropout)           (None, 50)                0

 dense_1 (Dense)             (None, 2)                 102

=================================================================
Total params: 1,264,177
Trainable params: 1,264,177
Non-trainable params: 0
_________________________________________________________________
None
```

Apprentissage

```
epochs = 10
ThisModel = KerasModel.fit(X_train, y_train, epochs=epochs,batch_size=64,verbose=1)
```

```
Epoch 1/10
2/2 [==============================] - 6s 2s/step - loss: 198.0886 - accuracy: 0.4167
Epoch 2/10
2/2 [==============================] - 3s 2s/step - loss: 255.9957 - accuracy: 0.5250
Epoch 3/10
2/2 [==============================] - 4s 2s/step - loss: 142.0237 - accuracy: 0.5500
Epoch 4/10
2/2 [==============================] - 4s 2s/step - loss: 96.2295 - accuracy: 0.5083
Epoch 5/10
2/2 [==============================] - 3s 2s/step - loss: 36.3211 - accuracy: 0.5917
Epoch 6/10
2/2 [==============================] - 3s 2s/step - loss: 16.9353 - accuracy: 0.6333
Epoch 7/10
2/2 [==============================] - 4s 2s/step - loss: 3.2667 - accuracy: 0.7333
Epoch 8/10
2/2 [==============================] - 4s 2s/step - loss: 3.2639 - accuracy: 0.7000
Epoch 9/10
2/2 [==============================] - 3s 2s/step - loss: 1.2234 - accuracy: 0.7333
Epoch 10/10
2/2 [==============================] - 4s 2s/step - loss: 0.6010 - accuracy: 0.8167
```

Double-cliquez (ou appuyez sur Entrée) pour modifier

Mon accuracy est de 81,67 % lors de l'apprentissage est proche de 100% donc l'apprentissage est plutôt bon. Il est difficile pour la machine de trouver les similtudes car même si pour l'homme à l'oeil nu pour l'oiseau on peut imaginer qu'il y a un bec des fois cela ne se voit pas dansn l'image. Les différences telles que les ailes et le bac sont des différences qui ne sont pas faciles à détecter par la machine selon la position variante de l'oiseau. Cependant, nous pouvons voir que le loss est de 0,60 ce qui montre que le modèle a réussi à ajuster les poids et les biais de ses couches pour minimiser l'erreur sur les données d'entraînement.

```
ModelLoss, ModelAccuracy = KerasModel.evaluate(X_test, y_test)

print('Test Loss is {}'.format(ModelLoss))
print('Test Accuracy is {}'.format(ModelAccuracy ))
```

```
2/2 [==============================] - 1s 306ms/step - loss: 0.3105 - accuracy: 0.7667
Test Loss is 0.31054258346557617
Test Accuracy is 0.7666666507720947
```

Le test loss mesure l'erreur moyenne du modèle sur l'ensemble de données de test. Dans ce cas, le test loss est de 31,05%.

Le test accuracy mesure de la capacité du modèle à classer correctement les exemples de l'ensemble de données de test. Dans ce cas, le test accuracy est de 0,7667, donc le modèle a réussi à classifier correctement 76,67 % des exemples de l'ensemble de données de test.

Evaluation

```
from keras.metrics import Precision, Recall


accuracy, recall= KerasModel.evaluate(X_test, y_test)

print ('Test accuracy:', accuracy)
print ('Test recall', recall)
```

```
     2/2 [==============================] - 1s 446ms/step - loss: 0.3105 - accuracy: 0.7667
     Test accuracy: 0.31054258346557617
     Test recall 0.7666666507720947
```

Rappelons-le, le test accuracy mesure l'exactitude du modèle, c'est-à-dire le pourcentage de prédictions correctes sur l'ensemble des prédictions effectuées lors du test. Ici, le test accuracy est de 31,05%, ce qui signifie que le modèle a correctement prédit seulement 31,05% des résultats lors du test.

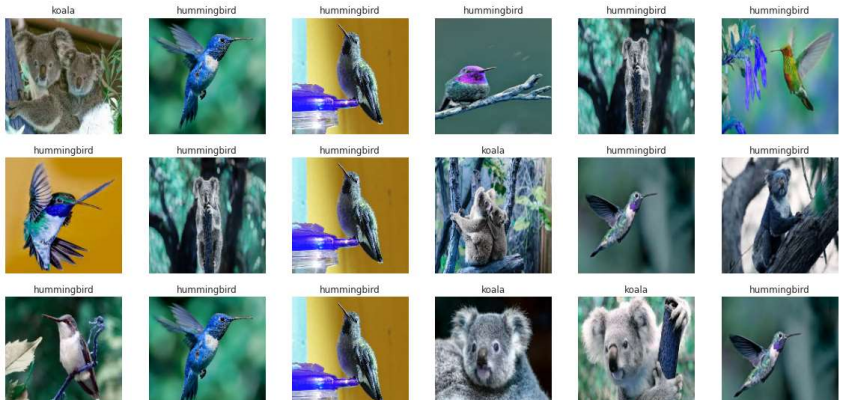
Le test recall est une mesure de la capacité du modèle à identifier tous les exemples positifs correctement identifiés par le modèle. Ici, le test recall est d'environ 0,76, ce qui signifie que le modèle a correctement identifié 76% des exemples positifs lors du test.

On voit que le modèle de classification binaire n'a pas bien fonctionné lors du test, car il n'a correctement prédit qu'un petit pourcentage des résultats et n'a identifié qu'environ la moitié des exemples positifs.

```
y_pred= KerasModel.predict(X_test)

print('Predictiion Shape is {}'.format(y_pred.shape))
```

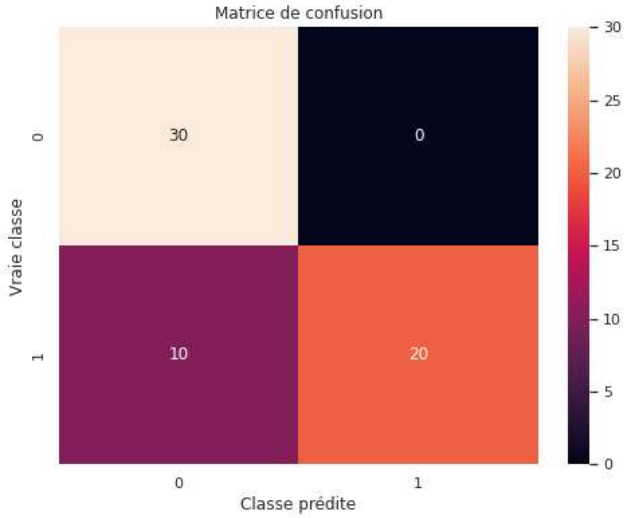
```
     2/2 [==============================] - 1s 349ms/step
     Predictiion Shape is (60, 2)
```

```
plt.figure(figsize=(20,20))
for n , i in enumerate(list(np.random.randint(0,len(X_test),36))) :
    plt.subplot(6,6,n+1)
    plt.imshow(X_test[i])
    plt.axis('off')
    plt.title(getcode(np.argmax(y_pred[i])))
```




```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

from sklearn.metrics import confusion_matrix

# Faire des prédictions sur les données de test
y_pred = KerasModel.predict(X_test)

# Convertir les prédictions en classe de sortie binaire (0 ou 1)
y_pred_classes = np.argmax(y_pred,axis = 1)

# Calculer la matrice de confusion
confusion_mtx = confusion_matrix(y_test, y_pred_classes)

# Afficher la matrice de confusion
plt.figure(figsize=(8,6))
sns.heatmap(confusion_mtx, annot=True, fmt="d");
plt.title("Matrice de confusion")
plt.ylabel('Vraie classe')
plt.xlabel('Classe prédite')
plt.show()
```

```
2/2 [==============================] - 2s 577ms/step
```



Le modèle a correctement prédit que l'observation appartient à la classe positive (20 fois sur les 30 fois qu'il fallait, soit environ 66%). De plus, le modèle a correctement prédit que l'observation appartient à la classe négative (30 fois sur les 30 soit 100% de réussite)