In [1]:
```
import keras
keras.__version__
import tensorflow as tf
tf.__version__
```

```
Using TensorFlow backend.
```

Out[1]:
```
'2.1.0'
```

In [2]:
```
# Ignore  the warnings
import warnings
warnings.filterwarnings('always')
warnings.filterwarnings('ignore')

# data visualisation and manipulation
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
import seaborn as sns

#configure
# sets matplotlib to inline and displays graphs below the corressponding cell.
%matplotlib inline
style.use('fivethirtyeight')
sns.set(style='whitegrid',color_codes=True)

#model selection
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score,precision_score,recall_score,confusion
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import LabelEncoder

#preprocess.
from keras.preprocessing.image import ImageDataGenerator

#dl libraraies
from keras import backend as K
from keras import regularizers
from keras.models import Sequential
from keras.models import Model
from keras.layers import Dense
from keras.optimizers import Adam,SGD,Adagrad,Adadelta,RMSprop
from keras.utils import to_categorical
from keras.callbacks import ReduceLROnPlateau
from keras.models import load_model
# specifically for cnn
from keras.layers import Dropout, Flatten,Activation
from keras.layers import Conv2D, MaxPooling2D, BatchNormalization
from keras.layers import InputLayer

import tensorflow as tf
import random as rn

# specifically for manipulating zipped images and getting numpy arrays of pixel v
import cv2
import numpy as np
from tqdm import tqdm
import os
from random import shuffle
from zipfile import ZipFile
from PIL import Image

from keras.applications.resnet50 import ResNet50
```

```
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
```

In [3]:
```
os.listdir('mask_dataset')
```

Out[3]:
```
['0', '1']
```

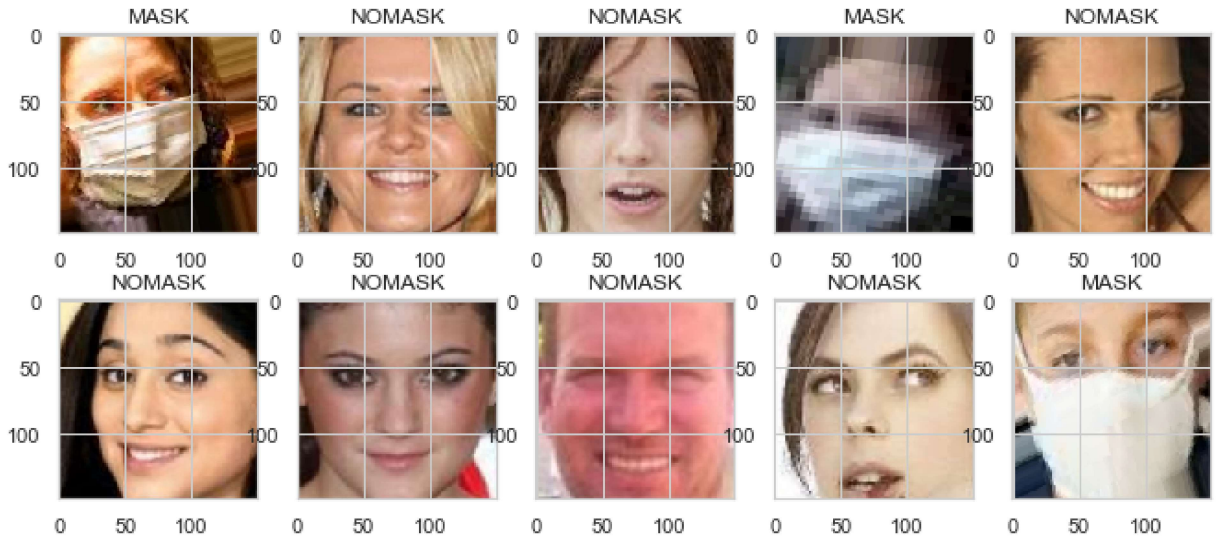
In [4]:
```
def assign_label(img,label):
    return label
```

In [5]:
```
def make_train_data(label,DIR):
    for img in tqdm(os.listdir(DIR)):
        path = os.path.join(DIR,img)
        img = cv2.imread(path,cv2.IMREAD_COLOR)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))

        X.append(np.array(img))
        Z.append(str(label))
```

In [6]:
```
X=[]
Z=[]
IMG_SIZE=150
NOMASK='mask_dataset/0'
MASK='mask_dataset/1'

make_train_data('NOMASK',NOMASK)
make_train_data('MASK',MASK)
```

```
100%|██████████| 509/509 [00:00<00:00, 732.22it/s]
100%|██████████| 483/483 [00:01<00:00, 369.23it/s]
```

In [7]:
```
fig,ax=plt.subplots(2,5)
plt.subplots_adjust(bottom=0.3, top=0.7, hspace=0)
fig.set_size_inches(10,10)

for i in range(2):
    for j in range (5):
        l=rn.randint(0,len(Z))
        ax[i,j].imshow(X[l][:,:,::-1])
        ax[i,j].set_title(Z[l])
        ax[i,j].set_aspect('equal')
```



In [8]:
```
le=LabelEncoder()
Y=le.fit_transform(Z)
Y=to_categorical(Y,2)
print(Y)
X=np.array(X)
#X=X/255

x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.25,random_state=1

np.random.seed(42)
rn.seed(42)
#tf.set_random_seed(42)
```

```
[[0. 1.]
 [0. 1.]
 [0. 1.]
 ...
 [1. 0.]
 [1. 0.]
 [1. 0.]]
```

In [9]:
```
base_model=ResNet50(include_top=False, weights='imagenet',input_shape=(150,150,3)
base_model.summary()
```

```
Model: "resnet50"
__________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
input_1 (InputLayer)            (None, 150, 150, 3)  0
__________________________________________________________________________________
conv1_pad (ZeroPadding2D)       (None, 156, 156, 3)  0           input_1[0]
[0]
__________________________________________________________________________________
conv1 (Conv2D)                  (None, 75, 75, 64)   9472        conv1_pad[0]
[0]
__________________________________________________________________________________
bn_conv1 (BatchNormalization)   (None, 75, 75, 64)   256         conv1[0][0]
__________________________________________________________________________________
```

In [10]:
```
model=Sequential()
model.add(base_model)
model.add(Dropout(0.20))
model.add(Dense(2048,activation='relu'))
model.add(Dense(1024,activation='relu'))
model.add(Dense(512,activation='relu'))
model.add(Dense(2,activation='softmax'))
```

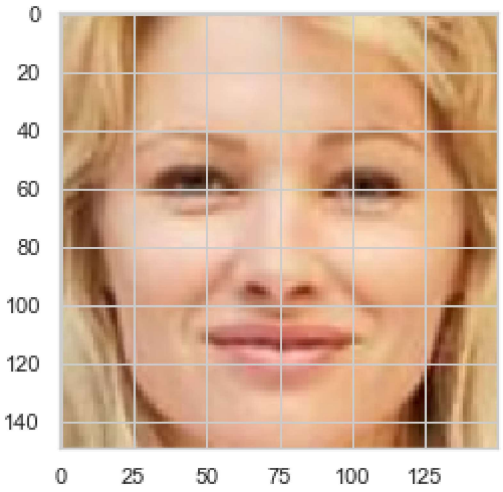
In [11]:
```
epochs=100
batch_size=128
red_lr=ReduceLROnPlateau(monitor='val_acc', factor=0.1, min_delta=0.0001, patienc
base_model.trainable=True # setting the VGG model to be trainable.
model.compile(optimizer=Adam(lr=1e-5),loss='categorical_crossentropy',metrics=['a
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Model)             (None, 2048)              23587712
_________________________________________________________________
dropout_1 (Dropout)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 2048)              4196352
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              2098176
_________________________________________________________________
dense_3 (Dense)              (None, 512)               524800
_________________________________________________________________
dense_4 (Dense)              (None, 2)                 1026
=================================================================
Total params: 30,408,066
Trainable params: 30,354,946
Non-trainable params: 53,120
_________________________________________________________________
```

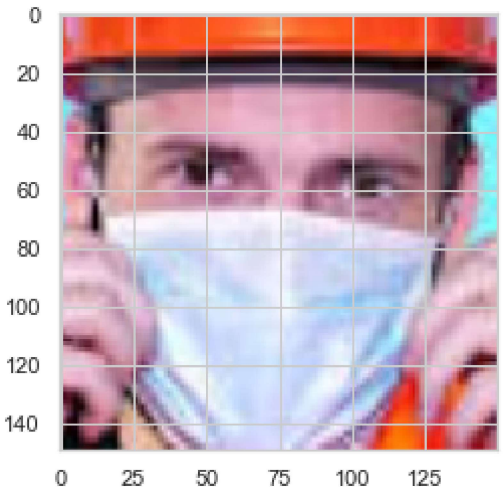
In [12]:
```
from keras.models import load_model
model = load_model('model.h5')
```

In [20]:
```
path="mask_check/"
files=os.listdir(path)
for i in files:
    X=cv2.imread(path+i)
    X=cv2.resize(X,(150,150))
    plt.figure()
    plt.imshow(X[:,:,::-1])
    plt.show()  # display it


    X = np.array(X)
   # X = X/255
    X = np.expand_dims(X, axis=0)

    #print(np.round(model.predict(X)))
    fire=int(np.round(model.predict(X))[0][0])
    if fire==0:
        print("Absence of mask detected")


        fromaddr = "sreedhar.uniq@gmail.com"
        toaddr = "sreedharsachu1401@gmail.com"

        # instance of MIMEMultipart
        msg = MIMEMultipart()

        # storing the senders email address
        msg['From'] = fromaddr

        # storing the receivers email address
        msg['To'] = toaddr

        # storing the subject
        msg['Subject'] = "no mask"

        # string to store the body of the mail
        body = "Not wearing any mask"

        # attach the body with the msg instance
        msg.attach(MIMEText(body, 'plain'))

        # open the file to be sent
        filename = str(i)+".jpg"
        att=path+i
        attachment = open(att, "rb")

        # instance of MIMEBase and named as p
        p = MIMEBase('application', 'octet-stream')

        # To change the payload into encoded form
        p.set_payload((attachment).read())

        # encode into base64
        encoders.encode_base64(p)

        p.add_header('Content-Disposition', "attachment; filename= %s" % filename
```

```
        # attach the instance 'p' to instance 'msg'
        msg.attach(p)

        # creates SMTP session
        s = smtplib.SMTP('smtp.gmail.com', 587)

        # start TLS for security
        s.starttls()

        # Authentication
        s.login("sreedhar.uniq@gmail.com", "1017875reg")

        # Converts the Multipart msg into a string
        text = msg.as_string()

        # sending the mail
        s.sendmail(fromaddr, toaddr, text)

        # terminating the session
        s.quit()
    else:
        print("mask present")
```




```
Absence of mask detected
```



```
mask present
```

In [ ]:

In [ ]: